

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-057032

(43)Date of publication of application : 25.02.2000

(51)Int.Cl.

G06F 12/00  
G06F 17/30

(21)Application number : 11-101694

(71)Applicant : MITSUBISHI ELECTRIC INF  
TECHNOL CENTER AMERICA INC

(22)Date of filing : 08.04.1999

(72)Inventor : PENG LUOSCHENG

(30)Priority

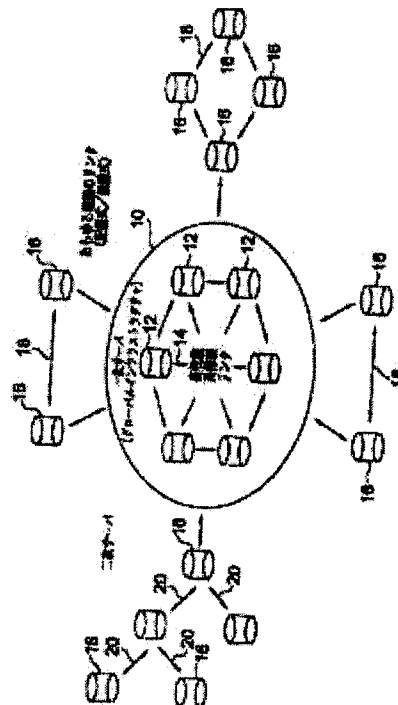
Priority number : 98 110748    Priority date : 03.07.1998    Priority country : US

(54) OBJECT SYNCHRONIZING GENERAL-PURPOSE SYSTEM USING PLURAL SERVERS, OBJECT SYNCHRONIZING SYSTEM AND METHOD USING TWO SERVERS, GENERAL-PURPOSE SYNCHRONIZING SYSTEM IN WHICH OBJECT EXISTING IN TWO SERVERS ARE SYNCHRONIZED REGARDLESS OF TYPE OF OBJECT AND METHOD FOR DETECTING AND ELIMINATING COMPETITION IN SYNCHRONIZATION OF OBJECTS IN TWO SERVERS

(57)Abstract:

PROBLEM TO BE SOLVED: To secure the synchronization of objects by means of plural servers by securing the synchronization between the object of a secondary server and that of a primary server to which the secondary server can be connected at the secondary server when a link is established between the said two objects.

SOLUTION: The primary servers 12 of a network 10 are connected to each other via a high performance link 14 which has higher reliability than a radio link in general. Many secondary servers 16 are linked (18) to each other in a pier-to-pier relation or linked (20) to each other in a hierarchical system. Then the data are automatically synchronized with each other among plural primary servers 12, and a 1st server 16 is connected to one of servers 12 via many servers 16 and a link. The object of the 1st server 16 is synchronized with the object of any one of servers 12 to which the server 16 can be connected at the server 16 when a link is established between these two servers.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-57032

(P2000-57032A)

(43) 公開日 平成12年2月25日 (2000.2.25)

(51) Int.Cl.<sup>7</sup>

G 0 6 F 12/00  
17/30

識別記号

5 3 3

F I

G 0 6 F 12/00  
15/401

テーマコード\* (参考)

5 3 3 J  
3 4 0 B

審査請求 未請求 請求項の数16 OL (全 24 頁)

(21) 出願番号

特願平11-101694

(22) 出願日

平成11年4月8日 (1999.4.8)

(31) 優先権主張番号

09/110748

(32) 優先日

平成10年7月3日 (1998.7.3)

(33) 優先権主張国

米国 (US)

(71) 出願人 597067574

ミツビシ・エレクトリック・インフォメイ  
ション・テクノロジー・センター・アメリ  
カ・インコーポレイテッド

MITSUBISHI ELECTRIC  
INFORMATION TECHNO  
LOGY CENTER AMERIC  
A, INC.

アメリカ合衆国、マサチューセッツ州、ケ  
ンブリッジ、ブロードウェイ 201

(74) 代理人 100057874

弁理士 曾我 道照 (外6名)

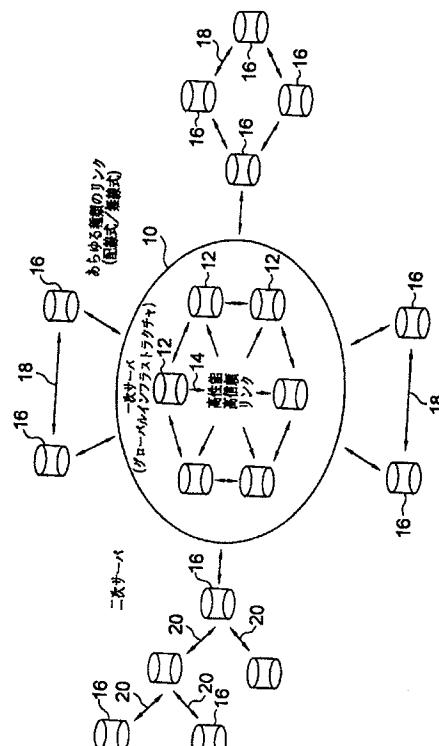
最終頁に続く

(54) 【発明の名称】 複数のサーバでオブジェクトを同期させる汎用システム、2つのサーバでオブジェクトを同期させるシステム、2つのサーバでオブジェクトを同期させる方法、2つのサーバに存在するオブジ

(57) 【要約】

【課題】 広域モバイルコンピューティングに適応し、サーバを同期させ、同時にプロセスをより効率的にするための汎用システムを提供する。

【解決手段】 二次サーバが概して信頼性の低いサーバによりリンクされた同期プロセスのバックボーンを形成する、一次サーバの高性能高信頼性リンクを用いたネットワークを備え、モバイル・コンピュータからの同期をクライアント/サーバ・モード及びピア・ツー・ピアの双方で行うことができ、二次サーバのいかなるトポロジをもサポートすることができるようにする。一次サーバは自動的にかつ頻繁に同期をとるが、二次サーバの同期は、意図しない同期を避けるユーザの制御下で行われる。また、要約バージョンベクトルを使用して、オブジェクト毎にバージョンベクトルを交換する必要を無くすことにより、転送されるデータの量を最小にする。



## 【特許請求の範囲】

【請求項1】 広域モバイルコンピューティングに適応する複数のサーバでオブジェクトを同期させる汎用システムであって、

複数の一次サーバのネットワークと、

上記複数の一次サーバを接続する高性能高信頼性リンクと、

上記複数の一次サーバにおいてデータを自動的に同期させる手段と、

多数の二次サーバと、

リンクによって第1の二次サーバを上記複数の一次サーバの1つに接続する手段と、

二次サーバのオブジェクトを、二次サーバが接続され得るいずれかの一次サーバのオブジェクトと、それらの間にリンクが確立した時に、二次サーバにおいて同期させる手段とを含むことを特徴とする複数のサーバでオブジェクトを同期させる汎用システム。

【請求項2】 第2の二次サーバと、上記第1及び第2の二次サーバ間の同期リンクとをさらに具備し、クライアント／サーバ及びピア・ツー・ピア(peer to peer)の同期モードをサポートすることを特徴とする請求項1記載の複数のサーバでオブジェクトを同期させる汎用システム。

【請求項3】 広域モバイルコンピューティングに適応する複数のサーバでオブジェクトを同期させる汎用システムであって、

複数の一次サーバのネットワークと、

上記一次サーバを接続する高性能高信頼性リンクと、

上記一次サーバにおいてオブジェクトを自動的に同期させる手段と、

多数の二次サーバと、

リンクによって、二次サーバを上記複数の一次サーバの1つに接続する手段と、

二次サーバにおいて二次サーバによって開始され、二次サーバのオブジェクトを、二次サーバが接続され得るいずれかの一次サーバのオブジェクトと同期させる手段とを具備し、

上記二次サーバ及び上記一次サーバの間の上記リンクの信頼性が低い場合でも、上記二次サーバが、接続され得る一次サーバとの間に高信頼性リンクを確立することが可能であると判断した時、同期が起こることを特徴とする複数のサーバでオブジェクトを同期させる汎用システム。

【請求項4】 上記同期させる手段は、上記二次サーバにおいて、上記リンクの信頼性を確認し、十分に信頼性の高いリンクを確立することができる場合にのみ、同期を開始する確認手段を含むことを特徴とする請求項3記載の複数のサーバでオブジェクトを同期させる汎用システム。

【請求項5】 上記確認手段は 上記リンクにより情報

を転送するための帯域幅を決定する手段を含むことを特徴とする請求項4記載の複数のサーバでオブジェクトを同期させる汎用システム。

【請求項6】 上記確認手段は、上記二次サーバで使用可能なリソースを確認する手段を含むことを特徴とする請求項4記載の複数のサーバでオブジェクトを同期させる汎用システム。

【請求項7】 上記同期させる手段は、上記二次サーバで、上記同期を手動で開始する手段を有することを特徴とする請求項3記載の複数のサーバでオブジェクトを同期させる汎用システム。

【請求項8】 2つのサーバでオブジェクトを同期させるシステムであって、

2つの別々のロケーションにあるサーバと、

上記サーバをリンクするネットワークと、

上記サーバの1つにおいて、オブジェクト全体を基準とするか、又は差分を基準として、上記2つのサーバでオブジェクトを同期させる手段と、

上記同期させる手段に、オブジェクト全体の同期と差分同期とを切換えさせる手段とを含むことを特徴とする2つのサーバでオブジェクトを同期させるシステム。

【請求項9】 上記同期させる手段に切換えさせる手段は、上記サーバの1つにおいて使用可能なリソースを確認し、サーバのリソースが差分同期をサポートすることができない場合に、オブジェクト全体の同期に切換える手段を有することを特徴とする請求項8記載の2つのサーバでオブジェクトを同期させるシステム。

【請求項10】 個々のオブジェクトのバージョンベクトルを交換する必要を無くすことにより、転送されるデータの量を最小にするように2つのサーバでオブジェクトを同期させる方法であって、

上記サーバの各々に、オブジェクトを包含するよう適合されたオブジェクトコンテナを供給するステップと、サーバにおけるオブジェクトの各々の状態を要約する要約バージョンベクトルを各オブジェクトコンテナに供給するステップと、

第1のサーバから第2のサーバへ一つの要約バージョンベクトルのみを転送し、該第2のサーバにおけるオブジェクトに関連するバージョンベクトルが、該第1のサーバから受信した要約バージョンベクトルより新しいか又はそれと競合(conflict)する場合、あるいは、該第2のサーバにおける個々のオブジェクトの更新されたタイムスタンプが、該第1のサーバからの要約バージョンベクトルに関連したタイムスタンプより新しいか又はそれと競合する場合、即座にその第2のサーバから第1のサーバへ更新を返すことにより、同期を開始するステップとを含み、

上記要約バージョンベクトルは、上記オブジェクトコンテナの状態を要約すると共に、更新スタンプを有し、その更新スタンプの各々は 上記オブジェクトコンテナに

関連する識別子用のフィールドと、上記オブジェクトコンテンツがオブジェクトを生成し、変更し又は削除した最後の時刻に対応するタイムスタンプ用のフィールドとを有し、上記タイムスタンプは、オブジェクトが生成され、変更され又は削除された時にそのオブジェクトコンテンツによって生成されることを特徴とする2つのサーバでオブジェクトを同期させる方法。

【請求項11】 上記同期を開始するステップは、上記要約バージョンベクトル及び更新スタンプに基づいて差分同期を実行して差分更新を生成し、それによって、差分更新によって示される、変更されているオブジェクトの部分のみを転送することによって、差分同期を行うステップを含むことを特徴とする請求項10記載の2つのサーバでオブジェクトを同期させる方法。

【請求項12】 最新共通バージョンベクトルを生成し、上記最新共通バージョンベクトルを使用して、選択された差分更新を一掃するステップをさらに含むことを特徴とする請求項11記載の2つのサーバでオブジェクトを同期させる方法。

【請求項13】 影響されていないサーバからのデータを用いて、前の失敗の地点から同期を再開するステップであって、前の失敗の地点と影響されていないサーバの存在を、そのサーバに関連する要約バージョンベクトルによって決定するステップをさらに含むことを特徴とする請求項10記載の2つのサーバでオブジェクトを同期させる方法。

【請求項14】 第1のサーバから第2のサーバへの更新に対応するオブジェクトのバージョンベクトルと、該第2のサーバの要約バージョンベクトルとの両方を、該第2のサーバが該第1のサーバから更新を受信した直後に更新し、それによって、その第1のサーバの要約バージョンベクトルを該第2のサーバの更新されたバージョンベクトルと比較することにより、該第2のサーバによってすでに受信されている更新を再送することなく、同期を復元することができるようにし、それによってきめの細かい同期を提供するステップをさらに含むことを特徴とする請求項10記載の2つのサーバでオブジェクトを同期させる方法。

【請求項15】 2つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期する汎用同期システムであって、  
別々のロケーションで、オブジェクトのバージョンとそれに関連する更新ベクトルをそれぞれ有する複数のサーバと、

上記サーバをリンクするネットワークと、  
オブジェクトのフォーマットに関係なく標準フォーマットでオブジェクトの更新を抽出する手段、及び、上記標準フォーマットのオブジェクトの更新を利用して標準プロトコルに基づいて上記複数のサーバでオブジェクトを同期させる手段を含むサーバにおいて上記オブジェク

トのセマンティクスを各上記同期から切離す手段とを具備し、

同期をオブジェクトの形態に関係なく行うことができ、それによって異なるシステム間での同期が可能であることを特徴とする2つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期する汎用同期システム。

【請求項16】 第1のサーバから第2のサーバへ更新を伝搬し、2つのサーバでのオブジェクトの同期における競合を検出して解消する方法であって、

上記第1のサーバから上記第2のサーバへ要約バージョンベクトルを送信するステップと、

上記第1のサーバの要約バージョンベクトルを受信すると、上記第1のサーバに上記第2のサーバの要約バージョンベクトルを返信し、その後第2のサーバに存在し差分同期をサポートすることができるオブジェクトの識別子をすべて送信するステップと、

上記第2のサーバから要約バージョンベクトルと識別子を受信すると、第1のサーバにおいて、受信した識別子から、上記第1のサーバに存在しないオブジェクトに対応する識別子を決定するステップと、

上記第1のサーバにおいて、上記2つのサーバの要約バージョンベクトルの共通するバージョンベクトルを計算するステップと、

上記第1のサーバにおいて、差分同期をサポートすることができるオブジェクトであって、そのバージョンベクトルについて、対応する差分更新がすべて無いベースが、上記計算された共通のバージョンベクトルより新しいオブジェクトの識別子をすべて決定し、オブジェクトの第1及び第2の決定された識別子が、同期において差分同期を実現することができず、そのためにオブジェクト全体の同期に切換えられなければならないオブジェクトの識別子であるステップと、

オブジェクト全体の同期に切換えられなければならないオブジェクトの識別子を上記第2のサーバに送信するステップと、

上記第1のサーバから上記識別子を受信すると、上記第2のサーバにおいて、その2つのサーバの要約バージョンベクトルの共通するバージョンベクトルを計算するステップと、

上記第2のサーバにおいて、上記受信した識別子と異なるオブジェクトの識別子であり、差分同期をサポートすることができるオブジェクトであって、そのバージョンベクトルについて、対応する差分更新がすべて無いベースが、上記計算された共通のバージョンベクトルより新しいオブジェクトの識別子をすべて決定し、決定されたオブジェクトが、同期において差分同期を実現することができず、そのためにオブジェクト全体の同期に切換えられなければならないオブジェクトであるステップと、

上記第2のサーバから上記第1のサーバへ送信する場合

にオブジェクト全体で送信しなければならないオブジェクトの識別子のセットとして、上記決定された識別子に上記受信した識別子を加えるステップと、

上記第2のサーバにおいて、差分同期をサポートすることができないオブジェクト、及び、差分同期をサポートすることができ、その識別子が上記第2のサーバで決定された識別子のセットにあるオブジェクトを含む個々のオブジェクトのバージョンベクトルを、上記第1のサーバの要約バージョンベクトルと比較するステップと、  
オブジェクト全体である更新として、上記第1のサーバの要約バージョンベクトルより新しいか又はそれと競合するバージョンベクトルを有するオブジェクトを抽出するステップと、

上記第2のサーバにおいて、上記第2のサーバによって決定された識別子のセットに無い識別子を有する該第2のサーバのオブジェクトに与えられた個々の差分更新の更新スタンプを比較するステップと、

上記第1のサーバの要約バージョンベクトルより新しいか又はそれと競合する更新スタンプを有する差分更新を抽出するステップと、

上記抽出した更新のすべてを、対応するオブジェクトの識別子及びバージョンベクトル又は更新スタンプと共に、上記第2のサーバから上記第1のサーバへ、一貫した順序で送信するステップと、

上記第1のサーバへ上記更新を送信し終わると、上記第2のサーバにおいて、第2のサーバの最新共通先祖バージョンベクトルと差分更新の一部又はすべてをパージするステップと、

上記第2のサーバから、対応するオブジェクトの識別子及びバージョンベクトル又は更新スタンプと共に更新を受信すると、受信した更新がオブジェクト全体であるか差分更新であるかを判断するステップと、

上記受信した更新がオブジェクト全体である場合、受信したバージョンベクトルを上記第1のサーバの対応するオブジェクトのバージョンベクトルと比較し、上記受信した更新が、該第1のサーバにおけるオブジェクトのバージョンベクトルより古い場合又はそれと等しい場合、その受信したオブジェクトを破棄するステップと、

上記受信したバージョンベクトルが、上記第1のサーバのオブジェクトのバージョンベクトルより新しい場合、該第1のサーバのオブジェクトを上記受信したオブジェクトで置換えるか、あるいは、上記受信したバージョンベクトルが該第1のサーバのオブジェクトのバージョンベクトルと競合する場合、該第1のサーバのオブジェクトと上記受信したオブジェクトとの間に競合があることを識別し、該第1のサーバのオブジェクトにその競合を解消させるステップと、

上記受信した更新が差分更新である場合、上記受信した更新スタンプを上記第1のサーバの要約バージョンベクトルと比較するステップと

上記受信した更新スタンプが上記第1のサーバの要約バージョンベクトルより古い場合又はそれと等しい場合、上記受信した差分更新を捨てるステップと、

あるいは、上記第2のサーバの要約バージョンベクトルを、上記第1のサーバの対応するオブジェクトに与えられた該第1のサーバの各差分更新の更新スタンプと比較するステップと、

上記第2のサーバの要約バージョンベクトルより新しいか又はそれと競合する更新スタンプを有する差分更新をすべて抽出するステップと、

上記第1のサーバで差分更新が抽出されない場合、該第1のサーバのオブジェクトに上記受信した差分更新を与え、あるいは、上記受信した差分更新が該第1のサーバで抽出された差分更新と競合することを識別するステップと、

上記第1のサーバのオブジェクトに上記競合を解消させるステップと、

上記第2のサーバから上記更新を受信し終わると、上記第1のサーバの最新共通先祖バージョンベクトルを更新するステップと、

上記第1のサーバにおいて上記差分更新の一部又はすべてをパージするステップとを含むことを特徴とする2つのサーバでのオブジェクトの同期における競合を検出して解消する方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】この発明は、広域モバイルコンピューティングに適応する複数のサーバでオブジェクトを同期させる汎用システムに係るもので、複数のサーバでオブジェクトを同期させる汎用システム、2つのサーバでオブジェクトを同期させるシステム、2つのサーバでオブジェクトを同期させる方法、2つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期される汎用同期システム、2つのサーバでのオブジェクトの同期における競合を検出して解消する方法に関する。

【0002】

【従来の技術】ラップトップの形態のモバイルコンピューティングが急増し、あらゆる地点でデータのバージョンが生成されるに伴い、複数のロケーションで作成されるデータのバージョンを同期させる必要がある。このようなバージョンの同期は、個人がロケーションからロケーション、州から州、国から国へ移動するに従って、様々な異なったプラットフォームに互って行われなければならない。

【0003】複数のユーザが、文書又はデータの同じバージョンで作業をしている場合、従来は、文書を照合調整する様々な方法があった。あるユーザに対して別の文書が変更されたという事実を警告し、ユーザの判断で最新バージョンで更新するか、又は最新バージョンを拒否

するという、多くのシステムが提供されている。従来、バージョンが生成された時間を確認するためにログ・ファイルが使用されており、オブジェクトの複製（レプリカ）の間の競合を検出するために、長い間バージョンベクトルが利用されてきた。

【0004】理解されるように、解消すべき更新の競合の検出方法にはいくつかある。すなわち、James J. Kistler等による“Disconnected Operation in the CODA File System” (ACM Transactions on Computer Systems, 10(1), 1992)、Peter Reiher等による“Resolving File Conflicts in the Ficus File System” (USENIX Conference Proceedings, USENIX, June 1994)、Douglas B. Terry等による“Managing Update Conflicts in Bayou, a Weekly Connected Replicated Storage System” (Proceedings of the Fifth Symposium on Operating System Principles, ASM, December 1995) に述べられているような方法である。CODA及びFicusシステムは、D. Stott Parker等による“Detection of Mutual Inconsistency in Distributed Systems” (IEEE Transactions on Software Engineering 9(3), May 1993) において最初に提案されているように、オブジェクト・レプリカ間の競合を検出するためにバージョンベクトルを利用する。一方、Bayouシステムは、それ自身で競合を検出せず、アプリケーションに頼っている。

【0005】バージョンベクトルを利用して競合を検出する方法は、2つの基本的なパラダイムに発展してきている。1つのパラダイムでは、Ficusによって例示されているように、バージョンベクトルを利用して、個々のオブジェクト・レプリカの現在の状態について他のレプリカと比較して特徴を表す。CODAシステムによって推論することができるもう1つのパラダイムでは、バージョンベクトルを、オブジェクト・レプリカと、オブジェクトのセットを含むレプリケーション・ユニットとに適用している。

#### 【0006】

【発明が解決しようとする課題】CODAシステムは、ピア・ツー・ピア (peer to peer) 同期をサポートしていないファイル・レプリケーション・システムであることが理解されよう。それは、本質的に、2つのクライアントが互いに直接同期することができないクライアント／サーバ・システムである。例えば、2人のモバイルユーザが互いのデータを交換したい場合、彼らが前もって互いを知らなければ、これはピア・ツー・ピア同期でなければならない。

【0007】このような2つのクライアントがなんとかしてデータを同期させるためには、共通のサーバを紹介するという方法しかない。2つのクライアントが切断されると、リアルタイムに同期させることができなくなる。また、CODAシステムは、データを保存しCODAサーバと同期する方法を採用しているため、モバイルコン

ピューティングには適していない。CODAでは、サーバがオープンした時、クライアントは常に、使用可能なすべてのサーバ中でのファイルの最新バージョンを得ようとする。ファイルがクローズした時、クライアントは常に、使用可能なすべてのサーバと同期しようとする。

【0008】これは、大量のデータを転送するため、非常に費用がかかる。さらに、CODAでは、切断する前にファイル・アクセスがクライアントで保存されなかった場合、その切断されたオペレーションをサポートすることができない。これは、CODAでは、データが全体として複製されるのではなく、ファイルが逐一保存されるからである。このため、CODAシステムは、モバイルコンピューティングに適用しようとする場合、ファイルのオープン及びクローズの度に複数のサーバと並列同期をとる必要があるため、複雑である。

【0009】Ficusシステムは、クライアント／サーバ同期をサポートしていないファイル・レプリケーション・システムである。それは、本質的にピア・ツー・ピア・システムであり、それ自体でレプリケーション・サイト間のすべての同期が自動化されている。これは、モバイルコンピューティング環境では経済的ではない。そのような環境では、通信リンクが高価で信頼性がないという特徴を有しているため、Ficusシステムは望ましくない。

【0010】また、Ficusシステムでは、同期が自動的に行われるため、アプリケーションのセマンティクスが乱される可能性があり、それが原因で、このような場合にこのシステムが適用できなくなる。Ficusにおける同期は個々のファイルが保持するバージョンベクトルを使用することによって実現されるため、バージョンベクトルを各ファイルについて交換する必要がある。これは、無線通信には効率的でないと考えられる。例えば、ファイルのすべてのバージョンベクトルをサーバ間で転送しなければならないが、両方のサーバのファイルが同じである可能性がある。また、Ficusシステムは、同期プロセス中に全か無かの方式で全ファイルを送信するため、差分 (differential) 同期をサポートしていない。

【0011】最後に、Bayouシステムは、データベースに適用される書込みのログを利用して、差分同期をサポートするデータ・ベース・レプリケーション・システムである。しかしながら、Bayouシステムでは、単一の一次サーバを使用して書込みの順序及びコミットメントを包括的に最終決定しなければならない。単一の一次サーバがダウンすると、書込みの最終的なコミットメントが遅れることになる。このため、一次サーバがダウンした場合、書込みの最終的な順序を確立できない。これによって、他のサーバの、安定したデータベースをできるだけ早く参照するという能力が遅くなる。

【0012】さらに、Bayouシステムでは、アプリケーションが書込みを行う度に競合の検出及び解消のコー

ドを加える必要がある。この要件により問題が生じる。第1に、競合の検出はシステムが行うのではなくアプリケーションが行うため、アプリケーションによってはそれが負担となる可能性がある。第2に、競合の検出及び解消のためのコードは、各書込みと共に伝搬しなければならず、それによって、データ同期のコスト及びトラフィックが著しく増大する可能性がある。さらに、上述したシステムはすべて、異なるシステムに互って異なるタイプのデータを処理することができるほど柔軟ではない。

【0013】この発明は上述した点に鑑みてなされたもので、上述した問題をすべて対処することができ、特にモバイルコンピューティング環境のためのシステムに適応する複数のサーバでオブジェクトを同期させる汎用システム、2つのサーバでオブジェクトを同期させるシステム、2つのサーバでオブジェクトを同期させる方法、2つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期される汎用同期システム、2つのサーバでのオブジェクトの同期における競合を検出して解消する方法を提供するものである。

#### 【0014】

【課題を解決するための手段】この発明に係る複数のサーバでオブジェクトを同期させる汎用システムは、複数の一次サーバのネットワークと、上記複数の一次サーバを接続する高性能高信頼性リンクと、上記複数の一次サーバにおいてデータを自動的に同期させる手段と、多数の二次サーバと、リンクによって第1の二次サーバを上記複数の一次サーバの1つに接続する手段と、二次サーバのオブジェクトを、二次サーバが接続され得るいずれかの一次サーバのオブジェクトと、それらの間にリンクが確立した時に、二次サーバにおいて同期させる手段とを含むことを特徴とするものである。

【0015】また、第2の二次サーバと、上記第1及び第2の二次サーバ間の同期リンクとをさらに具備し、クライアント/サーバ及びピア・ツー・ピア(peer to peer)の同期モードをサポートすることを特徴とするものである。

【0016】また、この発明に係る複数のサーバでオブジェクトを同期させる汎用システムは、複数の一次サーバのネットワークと、上記一次サーバを接続する高性能高信頼性リンクと、上記一次サーバにおいてオブジェクトを自動的に同期させる手段と、多数の二次サーバと、リンクによって、二次サーバを上記複数の一次サーバの1つに接続する手段と、二次サーバにおいて二次サーバによって開始され、二次サーバのオブジェクトを、二次サーバが接続され得るいずれかの一次サーバのオブジェクトと同期させる手段とを具備し、上記二次サーバ及び上記一次サーバの間の上記リンクの信頼性が低い場合でも、上記二次サーバが、接続され得る一次サーバとの間に高信頼性リンクを確立する手段が可能であると判断し、

た時、同期が起こることを特徴とするものである。

【0017】また、上記同期させる手段は、上記二次サーバにおいて、上記リンクの信頼性を確認し、十分に信頼性の高いリンクを確立することができる場合にのみ、同期を開始する確認手段を含むことを特徴とするものである。

【0018】また、上記確認手段は、上記リンクにより情報を転送するための帯域幅を決定する手段を含むことを特徴とするものである。

10 【0019】また、上記確認手段は、上記二次サーバで使用可能なリソースを確認する手段を含むことを特徴とするものである。

【0020】また、上記同期させる手段は、上記二次サーバで、上記同期を手動で開始する手段を有することを特徴とするものである。

【0021】また、この発明に係る2つのサーバでオブジェクトを同期させるシステムは、2つの別々のロケーションにあるサーバと、上記サーバをリンクするネットワークと、上記サーバの1つにおいて、オブジェクト全体を基準とするか、又は差分を基準として、上記2つのサーバでオブジェクトを同期させる手段と、上記同期させる手段に、オブジェクト全体の同期と差分同期とを切り換えさせる手段とを含むことを特徴とするものである。

【0022】また、上記同期させる手段に切り換えさせる手段は、上記サーバの1つにおいて使用可能なリソースを確認し、サーバのリソースが差分同期をサポートすることができない場合に、オブジェクト全体の同期に切り換える手段を有することを特徴とするものである。

30 【0023】また、この発明に係る2つのサーバでオブジェクトを同期させる方法は、個々のオブジェクトのバージョンベクトルを交換する必要を無くすことにより、転送されるデータの量を最小にするように2つのサーバでオブジェクトを同期させる方法であって、上記サーバの各々に、オブジェクトを包含するよう適合されたオブジェクトコンテナを供給するステップと、サーバにおけるオブジェクトの各々の状態を要約する要約バージョンベクトルを各オブジェクトコンテナに供給するステップと、第1のサーバから第2のサーバへ一つの要約バージョンベクトルのみを転送し、該第2のサーバにおけるオブジェクトに関連するバージョンベクトルが、該第1のサーバから受信した要約バージョンベクトルより新しいか又はそれと競合(conflict)する場合、あるいは、該第2のサーバにおける個々のオブジェクトの更新されたタイムスタンプが、該第1のサーバからの要約バージョンベクトルに関連したタイムスタンプより新しいか又はそれと競合する場合、即座にその第2のサーバから第1のサーバへ更新を返すことにより、同期を開始するステップとを含み、上記要約バージョンベクトルは、上記オブジェクトコンテナの状態を要約すると共に、更新スタ

ンプを有し、その更新スタンプの各々は、上記オブジェクトコンテナに関連する識別子用のフィールドと、上記オブジェクトコンテナがオブジェクトを生成し、変更し又は削除した最後の時刻に対応するタイムスタンプ用のフィールドとを有し、上記タイムスタンプは、オブジェクトが生成され、変更され又は削除された時にそのオブジェクトコンテナによって生成されることを特徴とするものである。

【0024】また、上記同期を開始するステップは、上記要約バージョンベクトル及び更新スタンプに基づいて差分同期を実行して差分更新を生成し、それによって、差分更新によって示される、変更されているオブジェクトの部分のみを転送することによって、差分同期を行うステップを含むことを特徴とするものである。

【0025】また、最新共通バージョンベクトルを生成し、上記最新共通バージョンベクトルを使用して、選択された差分更新を一掃するステップをさらに含むことを特徴とするものである。

【0026】また、影響されていないサーバからのデータを用いて、前の失敗の地点から同期を再開するステップであって、前の失敗の地点と影響されていないサーバの存在を、そのサーバに関連する要約バージョンベクトルによって決定するステップをさらに含むことを特徴とするものである。

【0027】また、第1のサーバから第2のサーバへの更新に対応するオブジェクトのバージョンベクトルと、該第2のサーバの要約バージョンベクトルとの両方を、該第2のサーバが該第1のサーバから更新を受信した直後に更新し、それによって、その第1のサーバの要約バージョンベクトルを該第2のサーバの更新されたバージョンベクトルと比較することにより、該第2のサーバによってすでに受信されている更新を再送することなく、同期を復元することができるようにし、それによってきめの細かい同期を提供するステップをさらに含むことを特徴とするものである。

【0028】また、この発明に係る2つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期する汎用同期システムは、別々のロケーションで、オブジェクトのバージョンとそれに関連する更新ベクトルをそれぞれ有する複数のサーバと、上記サーバをリンクするネットワークと、オブジェクトのフォーマットに関係なく標準フォーマットでオブジェクトの更新を抽出する手段、及び、上記標準フォーマットのオブジェクトの更新を利用して標準プロトコルに基づいて上記複数のサーバでオブジェクトを同期させる手段を含む、サーバにおいて上記オブジェクトのセマンティクスを各上記同期から切離す手段とを具備し、同期をオブジェクトの形態に関係なく行うことができ、それによって異なるシステム間での同期が可能であることを特徴とするものである。

【0029】さらに、この発明に係る2つのサーバでのオブジェクトの同期における競合を検出して解消する方法は、第1のサーバから第2のサーバへ更新を伝搬し、2つのサーバでのオブジェクトの同期における競合を検出して解消する方法であって、上記第1のサーバから上記第2のサーバへ要約バージョンベクトルを送信するステップと、上記第1のサーバの要約バージョンベクトルを受信すると、上記第1のサーバに上記第2のサーバの要約バージョンベクトルを返信し、その後第2のサーバに存在し差分同期をサポートすることができるオブジェクトの識別子をすべて送信するステップと、上記第2のサーバから要約バージョンベクトルと識別子とを受信すると、第1のサーバにおいて、受信した識別子から、上記第1のサーバに存在しないオブジェクトに対応する識別子を決定するステップと、上記第1のサーバにおいて、上記2つのサーバの要約バージョンベクトルの共通するバージョンベクトルを計算するステップと、上記第1のサーバにおいて、差分同期をサポートすることができるオブジェクトであって、そのバージョンベクトルについて、対応する差分更新がすべて無いベースが、上記計算された共通のバージョンベクトルより新しいオブジェクトの識別子をすべて決定し、オブジェクトの第1及び第2の決定された識別子が、同期において差分同期を実現することができず、そのためにオブジェクト全体の同期に切換えられなければならないオブジェクトの識別子であるステップと、オブジェクト全体の同期に切換えられなければならないオブジェクトの識別子を上記第2のサーバに送信するステップと、上記第1のサーバから上記識別子を受信すると、上記第2のサーバにおいて、その2つのサーバの要約バージョンベクトルの共通するバージョンベクトルを計算するステップと、上記第2のサーバにおいて、上記受信した識別子と異なるオブジェクトの識別子であり、差分同期をサポートすることができるオブジェクトであって、そのバージョンベクトルについて、対応する差分更新がすべて無いベースが、上記計算された共通のバージョンベクトルより新しいオブジェクトの識別子をすべて決定し、決定されたオブジェクトが、同期において差分同期を実現することができず、そのためにオブジェクト全体の同期に切換えられなければならないオブジェクトであるステップと、上記第2のサーバから上記第1のサーバへ送信する場合にオブジェクト全体で送信しなければならないオブジェクトの識別子のセットとして、上記決定された識別子に上記受信した識別子を加えるステップと、上記第2のサーバにおいて、差分同期をサポートすることができないオブジェクト、及び、差分同期をサポートすることができ、その識別子が上記第2のサーバで決定された識別子のセットにあるオブジェクトを含む個々のオブジェクトのバージョンベクトルを、上記第1のサーバの要約バージョンベクトルと比較するステップと、オブジェクト全体である更



新として、上記第1のサーバの要約バージョンベクトルより新しいか又はそれと競合するバージョンベクトルを有するオブジェクトを抽出するステップと、上記第2のサーバにおいて、上記第2のサーバによって決定された識別子のセットに無い識別子を有する該第2のサーバのオブジェクトに与えられた個々の差分更新の更新スタンプを比較するステップと、上記第1のサーバの要約バージョンベクトルより新しいか又はそれと競合する更新スタンプを有する差分更新を抽出するステップと、上記抽出した更新のすべてを、対応するオブジェクトの識別子及びバージョンベクトル又は更新スタンプと共に、上記第2のサーバから上記第1のサーバへ、一貫した順序で送信するステップと、上記第1のサーバへ上記更新を送信し終わると、上記第2のサーバにおいて、第2のサーバの最新共通先祖バージョンベクトルと差分更新の一部又はすべてをパージするステップと、上記第2のサーバから、対応するオブジェクトの識別子及びバージョンベクトル又は更新スタンプと共に更新を受信すると、受信した更新がオブジェクト全体であるか差分更新であるかを判断するステップと、上記受信した更新がオブジェクト全体である場合、受信したバージョンベクトルを上記第1のサーバの対応するオブジェクトのバージョンベクトルと比較し、上記受信した更新が、該第1のサーバにおけるオブジェクトのバージョンベクトルより古いか又はそれと等しい場合、その受信したオブジェクトを破棄するステップと、上記受信したバージョンベクトルが、上記第1のサーバのオブジェクトのバージョンベクトルより新しい場合、該第1のサーバのオブジェクトを上記受信したオブジェクトで置換えるか、あるいは、上記受信したバージョンベクトルが該第1のサーバのオブジェクトのバージョンベクトルと競合する場合、該第1のサーバのオブジェクトと上記受信したオブジェクトとの間に競合があることを識別し、該第1のサーバのオブジェクトにその競合を解消させるステップと、上記受信した更新が差分更新である場合、上記受信した更新スタンプを上記第1のサーバの要約バージョンベクトルと比較するステップと、上記受信した更新スタンプが上記第1のサーバの要約バージョンベクトルより古いか又はそれと等しい場合、上記受信した差分更新を捨てるステップと、あるいは、上記第2のサーバの要約バージョンベクトルを、上記第1のサーバの対応するオブジェクトに与えられた該第1のサーバの各差分更新の更新スタンプと比較するステップと、上記第2のサーバの要約バージョンベクトルより新しいか又はそれと競合する更新スタンプを有する差分更新をすべて抽出するステップと、上記第1のサーバで差分更新が抽出されない場合、該第1のサーバのオブジェクトに上記受信した差分更新を与え、あるいは、上記受信した差分更新が該第1のサーバで抽出された差分更新と競合することを識別するステップと、上記第1のサーバのオブジェクトに上記競合を解消

させるステップと、上記第2のサーバから上記更新を受信し終わると、上記第1のサーバの最新共通先祖バージョンベクトルを更新するステップと、上記第1のサーバにおいて上記差分更新の一部又はすべてをパージするステップとを含むことを特徴とするものである。

#### 【0030】

【発明の実施の形態】以下、この発明を詳細に説明する。この発明によれば、多くの場合中央サーバ又はピア・ツー・ピア・システムいずれかへのリンクが信頼できないモバイルコンピューティング環境において、異なるサーバに格納されたオブジェクトのバージョンに対し信頼性のある同期をとるシステムが提供される。

【0031】このシステムは、単一の中央サーバ又はピア・ツー・ピア・サーバ・システムの代りに、二次サーバの同期をとるための高性能高信頼性リンクにリンクされた一次サーバのネットワークを用いる。複数の一次サーバを用いるため、データを、より優れたデータ整合性及び信頼性をもって、より広いエリアに分配することができる。モバイル・コンピュータを二次サーバとすることにより、これらの信頼性の低いサーバを、同期プロセスのバックボーンであるサーバから区別する。

【0032】一実施形態では、一次サーバは、一次サーバ間で優れたデータ整合性を維持するために、自動的にかつ頻繁に同期がとられる。一方、二次サーバからの信頼性の低い同期は、二次サーバから制御され、それによって通信コストを低減することができ、文書又は他のデータは、二次サーバでユーザが許可しなければ同期がとられない。これによって、ユーザは、文書のドラフトや、終了するまで同期がとられるべきではない未完成のソフトウェア又はプログラミングを用いて作業することができる。

【0033】重要な特徴として、本システムでは、要約(summarizing)バージョンベクトルを用いて、同期がとられている2つのオブジェクトに差があるか否かに関わらず、個々のオブジェクトについてバージョンベクトルを交換する必要を無くすことにより、同期プロセス中に転送されるデータの量を最小限にする。

【0034】ここで用いる「要約バージョンベクトル」という語は、サーバでオブジェクトコンテナの状態を要約するフィールドを有するベクトルを意味する。各要約バージョンベクトルは、更新スタンプのベクトルである。各更新スタンプは、関連するオブジェクトコンテナの識別子についてのフィールドと、関連するタイムスタンプについてのフィールドとを有している。

【0035】オブジェクトが同じでない場合、従来技術では、第1のロケーションにある各オブジェクトについてのバージョンベクトルが第2のロケーションに送信される時、対応する更新が第1のロケーションに返信されていた。このため、不要なデータが転送されるという状態が発生していた。

【0036】本システムでは、単一の要約バージョンベクトルが第1のロケーションから第2のロケーションに送信され、その後全ての更新が即座に第1のロケーションに返される。一実施形態では、要約バージョンベクトルが第2のロケーションで調べられ、第1のロケーションに、オブジェクト全体を送信する必要があるか、又はオブジェクトの一部を送信する必要があるかが判断される。これは、バージョンベクトルをオブジェクト全体に対して定義するか、又はバージョンベクトルを、オブジェクトのベース及びその差分更新各々についての更新スタンプに対して定義するかによって行われる。

【0037】ここで、オブジェクトのベースとは、差分更新の影響を受けた変化がまったく無い初期形態にあるオブジェクトを言う。

【0038】次に、受信した要約バージョンベクトルは、個々のオブジェクトの上記バージョンベクトル、又は個々の差分更新についての更新スタンプのいずれかと比較される。上述したように、差分更新の更新スタンプは、それを生成したオブジェクトコンテナの識別子と、それが生成された時のタイムスタンプからなる。

【0039】オブジェクト全体又は差分更新はすべて、それらのバージョンベクトル又は更新されたタイムスタンプが、受信した要約バージョンベクトルより新しいか、もしくはそれと競合する場合においてのみ、第1のロケーションに送信される。これにより、従来技術のシステムに関連する不要なバージョンベクトルの転送を無くすることができる。このため、要約バージョンベクトル・システムにより、個々のオブジェクトに関するバージョンベクトルを繰り返して転送することがなくなる。

【0040】要約バージョンベクトルの他の用途は、以下の通りである。なお、従来、同期システムは全か無かの基準で動作しており、それは、同期が完了前に失敗した場合に、すべてのデータが存在しなければならないことを意味していた。本システムでは、更新が第1のサーバから第2のサーバに送信される時、更新の受信が成功した直後に、対応するオブジェクトのバージョンベクトルと第2のサーバの要約バージョンベクトルの両方が更新される。

【0041】従って、同期が失敗した場合、第1のサーバの要約バージョンベクトルを第2のサーバの更新されたバージョンベクトルと比較することにより、前回の同期で第2のサーバから既に受信している更新を再送することなく、同期が復元される。このようなシステムでは、よりきめ細かい同期が提供され、また、そのため、このようなシステムはより耐障害性が優れている。

【0042】これは、本システムでは、転送されるデータのユニットが、差分更新（サイズが小さいためアトムと呼ばれる）であるという理由による。これは、転送データのユニット毎にオブジェクト全体を転送しなければならない従来技術のシステムとは異なっている。3つ以

上のサーバを有する場合、第3のサーバが、障害の時刻から再開の時刻までの間、第2のサーバと同期しているのならば、同期は、障害地点の後で第1のサーバ及び第2のサーバ間で復元される。これは、要約バージョンベクトルを用いることによって可能になる。

【0043】コンテナ内のオブジェクトは、例えば、文書、プログラム、リレーショナル・データベースのテーブルの行等、どのようなオブジェクトでもよく、そのため、本システムは汎用システムとなっていることが理解されよう。本システムは、様々なデータの形式に対しての同期プロセスを組込んでおり、それは、オブジェクトのセマンティクスを同期化から切離すことによって可能となっている。これは、データの形式に依存している既存のシステムとは対照的である。

【0044】この発明の重要な態様として、本システムでは、データのセマンティクスを同期化から切離しているため、Word文書、McDraw文書及びFrameMaker文書の間の同期化のように異なるシステム間の同期が可能である。これは、標準フォーマットで更新を抽出し、標準プロトコルに基づいて同期をとることによって行われる。

【0045】動作中に、あるフォーマットのあるロケーションでの更新が、標準プロトコルでの更新に変換される。標準プロトコルでの更新は、第2のロケーションに転送され、そこで第2のロケーションのオブジェクトのフォーマットに変換される。

【0046】さらに、本システムでは、データ転送を最小化するため、オブジェクトのバージョン間で変更されるデータのみに対して差分同期を使用する。この発明の固有の特徴として、本システムは、例えばローカル・メモリがいっぱいになった時、あるいはディスク・スペースが制限されている時に、オブジェクト全体の同期から差分同期へ切替える。さらに、あるロケーションのオブジェクトが非常に古い場合、差分同期が用いられているならば、オブジェクト全体または全部の同期に自動的にシフトすることによって、第2のロケーションのオブジェクト全体を更新するのが便利である。

【0047】この発明のさらなる特徴としては、モバイル・コンピュータからの同期を、クライアント／サーバ・モードにおいて二次サーバから一次サーバへ行うこともでき、また、ピア・ツー・ピア・システム又は階層システムのいずれかにおいて二次サーバ間で行うこともでき、これによってモバイル間同期が可能となることである。例えば、ある二次サーバがユーザのデスクトップ・コンピュータであり、他の二次サーバが二次的用途のためのモバイル装置である場合、この2つの二次サーバ間のデータ同期はクライアント／サーバ型であり、2つのモバイル装置のデータ同期（例えば、2人のセールスマンが所有し互いのセールス・データを交換する場合）は、ピア・ツー・ピア型で行う。本システムは、ピア・ツー・ピア構造及び階層構造の双方をサポートしている

結果、二次サーバのいかなるトポロジをもサポートする。

【0048】サーバは、しばしば、選択された数のサーバからのデータのみに関わるため、システム内のすべてのサーバと同期をとる必要はない。多くのサーバを有し、同期させたいデータを有するのはその一部のみであるようなシステムでは、すべてのオブジェクト及びすべての更新を追跡しようとする、メモリが急速に使い果たされる。

【0049】この発明の一実施形態でこの問題を解決するために、選択されたサーバのグループにおいて、更新及びバージョンの変更を選択的にパージするために、最新共通先祖バージョンベクトルを利用する。そのような更新及びバージョンの変更は、この最新共通先祖バージョンベクトルより古い又はそれと等しいものである。また、上記パージは、問題のサーバ、例えば選択されたサーバのグループに伝搬された差分更新又は削除されたオブジェクトをパージすることによって行う。

【0050】理想的には、システム内のすべてのサーバが更新を受信した場合においてのみ、サーバが差分更新又は削除されたオブジェクトをパージするべきである。しかしながら、サーバ数、特に二次サーバ数は動的に変化し、非常に長い間システム全体から切断されるサーバもあるため、すべてのサーバが更新を受信した後のみ更新をパージすることは実用的ではなく、メモリ又はディスク・スペースが、実際に更新をパージすることが可能となる前に、使い果たされてしまうことがある。

【0051】従って、本システムでは、サーバが、サーバのセットを選択して、選択されたサーバのすべてに更新が伝搬され終わった場合に、更新をパージすることができる。さらに、選択されたサーバは、予め設定された制限時間を超えて、選択しているサーバと同期していない場合、更新をパージする際に考慮の対象外とすることができる。

【0052】本システムでは、この更新のパージを、最新共通先祖バージョンベクトルを利用して行う。サーバは、選択されたサーバのいずれかと同期する場合、選択されたサーバのバージョンベクトルをローカルに記録する。そして、最新共通先祖バージョンベクトルは、それ自身のバージョンベクトルと選択されたすべてのサーバのバージョンベクトルとから計算される。

【0053】それらの更新スタンプ又はバージョンベクトルが、計算された最新共通バージョンベクトルより新しいか又はそれと等しい場合、すべての差分更新又は削除されたオブジェクトの情報がパージされる。以下に示す同期アルゴリズムは、要約バージョンベクトル及び更新スタンプの双方を用いて、転送されるデータを最小化する。同期中には、本アルゴリズムは、サーバからサーバへの更新の伝搬を実現するために、以下にリストしたステップを実行する。

【0054】第1のサーバは、その要約バージョンベクトルを第2のサーバに送信する。第2のサーバは、第1のサーバの要約バージョンベクトルを受信すると、自身の要約バージョンベクトルを第1のサーバに送信し、その後第2のサーバ内に存在し、差分同期をサポートすることができるすべてのオブジェクトの識別子を送信する。

【0055】第1のサーバは、第2のサーバから要約バージョンベクトルと識別子を受信すると、オブジェクト全体として受信し第2のサーバに送信する必要があるオブジェクトの識別子をすべて見つけ出す。

【0056】このステップのサブ・ステップは以下の通りである。

a) 第2のサーバ内に存在するが第1のサーバ内には存在しないすべてのオブジェクトを探し出し、それらの識別子を、第2のサーバから受信する必要があるオブジェクトのリストに加える。これは、受信したオブジェクトの識別子で第1のサーバ内に存在するオブジェクトの識別子と一致するものがあるか否かをチェックすることによって行う。

【0057】b) 両方のサーバに存在し、差分同期をサポートすることができるすべてのオブジェクトを探し出す。そして、これらのオブジェクトのベース・バージョンベクトルで2つのサーバの共通バージョンベクトルより新しいものがあるか否かをチェックする。ここで、オブジェクトのベース・バージョンベクトルとは、差分更新がまったくないオブジェクトのバージョンベクトルのことを言い、共通バージョンベクトルとは、2つのサーバの要約バージョンベクトルがそこから分岐した状態を反映するバージョンベクトルのことを言う。また、チェックをパスしたオブジェクトの識別子はすべて、第2のサーバから受信する必要があるオブジェクトのリストに加えられる。

【0058】c) サブ・ステップa) 及びb) から得られる識別子をすべて第2のサーバに送信する。

【0059】第2のサーバは、第1のサーバから識別子を受信すると、その最新共通先祖バージョンベクトルを第1のサーバの要約バージョンベクトルと比較する。第1のサーバの要約バージョンベクトルが第2のサーバの最新共通先祖バージョンベクトルより古い場合、それが含む個々のオブジェクト全体をすべて抽出する。そうでなければ、その要約バージョンベクトルと第1のサーバの要約バージョンベクトルとの比較を続ける。

【0060】第2のサーバの要約バージョンベクトルが第1のサーバの要約バージョンベクトルより新しい場合、第2のサーバは、その中に存在しているすべてのオブジェクトをスキャンする。オブジェクトの識別子が第1のサーバから受信した識別子のリストにある場合、それは全体として第1のサーバに送信される。オブジェクトが差分同期をサポートせず そのバージョンベクトル

が第1のサーバの要約バージョンベクトルより新しいか又はそれと競合するため、そのオブジェクトが全体として送信する必要のあるオブジェクトである場合、このオブジェクトもまた、第1のサーバに送信される。

【0061】オブジェクトが差分同期をサポートするオブジェクトであり、そのベース・バージョンベクトルが第1のサーバの要約バージョンベクトルより新しい場合、このオブジェクトもまた、全体として第1のサーバに送信される。最後に、オブジェクトが差分同期をサポートするオブジェクトであり、その更新のスタンプが第1のサーバの要約バージョンベクトルより新しいか又はそれと競合する場合、これらの更新は、差分更新として第1のサーバに送信される。送信されているオブジェクト又は更新のすべての順序は、バージョンベクトルの順序付け又は関連する更新スタンプによってまず決定される。順序付けに矛盾がある場合、二次的な順序付けプロセスが開始する。

【0062】ステップ4が終了した後、第2のサーバは、第1のサーバが第2のサーバが選択したサーバのうちの1つである場合、以前に格納していた第1のサーバの要約バージョンベクトルを格納又は更新する。また、選択されたサーバの要約バージョンベクトルがすべて第2のサーバに格納された場合、その最新共通先祖バージョンベクトルを再計算する。また、それによって、バージョンベクトル又は更新スタンプが最新共通先祖バージョンベクトルより古い又はそれと等しいすべての削除されたオブジェクトの情報又は差分更新を一掃する。

【0063】第1のサーバは、第2のサーバから各オブジェクト及び更新を受信すると、以下のように情報を更新する。

a) 受信したオブジェクト又は更新が、第1のサーバ内の対応するオブジェクトのバージョンベクトルより古い又はそれと等しいバージョンベクトル又はタイムスタンプを有する場合、このオブジェクト又は更新は、破棄される。

【0064】b) オブジェクト全体が受信され、そのバージョンベクトルが第1のサーバ内の対応するオブジェクトのバージョンベクトルより新しい場合、第1のサーバ内のオブジェクトは、受信されたオブジェクトに置換えられ、それによって、オブジェクトのバージョンベクトル及び第1のサーバの要約バージョンベクトルもまた更新される。

【0065】c) オブジェクト全体が受信され、そのバージョンベクトルが第1のサーバ内の対応するオブジェクトのバージョンベクトルと競合している場合、オブジェクトの2つのバージョンは、第1のサーバ内のオブジェクトについての照合調整メソッドを呼出すことによって、アーギュメントとして受信されたオブジェクトと調合調整される。その結果、第1のサーバ内のオブジェクトを置換えるために新たなオブジェクトが生成され、そ

れによって、新たなオブジェクトについての新たなバージョンベクトルが生成されて、第1のサーバの要約バージョンベクトルが更新される。

【0066】d) 差分更新が受信され、その更新スタンプ及び受信した第2のサーバの要約バージョンベクトルが、その差分更新がまだ第1のサーバ内に無く、第1のサーバにおいて対応するオブジェクトの差分更新後に生成されたことを示す場合、受信された更新は、第1のサーバ内のオブジェクトの差分更新のトップに加えられる。このように、第1のサーバの要約バージョンベクトルは、更新される。

【0067】e) 差分更新が受信され、その更新スタンプ及び受信した第2のサーバの要約バージョンベクトルが、差分更新がまだ第1のサーバ内に無く、第1のサーバにおいて対応するオブジェクトの差分更新後に生成されなかったことを示す場合、2つのサーバの共通する状態の後で生成された第1のサーバ内の差分更新が抽出され、抽出された差分更新及び受信した差分更新の両方が、それらの間の競合を照合調整するために、第1のサーバ内のオブジェクトに関する照合調整メソッドに対し、アーギュメントとして渡される。その結果、第1のサーバ内のオブジェクトは、新たなオブジェクトに置換えられるか、又は新たな更新が第1のサーバ内のオブジェクトの差分更新に加えられる。

【0068】このようにして、新たなオブジェクトの新たなバージョンベクトルか又は新たな更新についての更新スタンプが生成され、第1のサーバの要約バージョンベクトルが更新される。

【0069】ステップ6の終了後、第2のサーバが第1のサーバが選択したサーバのうちの1つである場合、第1のサーバは、以前に格納していた第2のサーバの要約バージョンベクトルを格納又は更新する。また、選択されたサーバの要約バージョンベクトルがすべて第1のサーバ内に格納されている場合、その最新共通先祖バージョンベクトルを再計算し、そのようにして、バージョンベクトル又は更新スタンプが最新共通先祖バージョンベクトルより古い又はそれと等しい削除されたオブジェクトの情報又は差分更新をすべてバースする。

【0070】ステップ7の終了後、第2のサーバから第1のサーバへの更新伝搬が完了する。両方向の更新伝搬が必要である場合、第1のサーバを第2のサーバで置換え、またその逆を行うことにより、ステップ1)からステップ7)を繰り返すことができる。

【0071】上記説明から分かるように、本アルゴリズムには、既存のアルゴリズムと比較して2つの利点がある。第1に、サーバからサーバへ、オブジェクト毎のバージョンベクトルを転送しなくてよい。第2に、あるサーバには取込まれているが他のサーバには取込まれていない差分更新を転送することができる。本アルゴリズムの更なる利点は、よりきめの細かい同期を提供し、か

つ、より耐障害性に優れているということである。

【0072】要約すると、広域モバイルコンピューティングに適応し、同時に同期プロセスをより効率的にするサーバを同期させるための汎用システムを提供する。第1に、高性能高信頼性リンクを有する複数の一次サーバを導入することにより、優れたデータ可用性及び拡張性が確保になる。第2に、二次サーバを導入することにより、本システムが、信頼性の低い高価なリンクを使用するモバイルコンピューティングに適するようになる。また、二次サーバによって同期を制御することにより、同期を経済的に行うことができる。さらに、自動的な同期を行うことができないアプリケーションがあるため、意図しない同期を防ぐ同期のユーザ制御がクリティカルである。

【0073】この発明の第2の態様は、要約バージョンベクトルを使用することである。この要約バージョンベクトルを使用して、個々のオブジェクトのバージョンベクトルを交換する必要をなくすことによって、同期プロセスにおいて転送されるデータの量を最小限にする。

【0074】また、個々の差分更新に関連した更新スタンプを用いて動作して差分同期を可能にすることにより、転送されるデータをさらに減らすことができる。サーバの要約バージョンベクトル及びその選択されたサーバのセットの要約バージョンベクトルから、最新共通バージョンベクトルを構成することができ、それを用いてサーバ上の差分更新を減らすことができる。

【0075】さらに、要約バージョンベクトルによって、影響を受けていないサーバからのデータを用いて、以前の失敗の地点から同期を再開することができる。また、転送されるデータのATOMとしての差分更新を可能にすることにより、きめの細かい同期をサポートする。

【0076】この発明の重要な特徴として、システムは、オブジェクト全体の同期と差分同期とを自動的に切替える。例えば、メモリ管理において、差分処理がメモリに対して集中する性質を有するために、差分同期からオブジェクト全体の同期へ切替えることがしばしば望ましい。また、オブジェクト全体の同期を行っている場合、転送されるデータを最少にするために差分同期に切替えることが有益な場合もある。

【0077】最後に、これも重要なことであるが、本システムでは、データのセマンティクスが同期化から切離しているため、本システムにより、異なるシステム間の同期が可能となり、これによりシステムがデータのフォーマットに関して汎用的なものとなる。

【0078】この発明のこれらの特徴及び他の特徴は、図面と共に詳細な説明を参照してより理解されるであろう。

【0079】図1では、一次及び二次サーバを有するシステム・アーキテクチャを示す。ここでは、一次サーバ12のネットワーク10が、一般に無線リンクより有線

いに信頼性の高い高性能リンク14を利用して、相互接続されている。高性能リンクを使用する目的は、同期保全性を確保にすることと、二次サーバの一次サーバへの接続を可能にすることである。

【0080】図示しているように、多数の二次サーバ16が、18に示すようにピア・ツー・ピア関係で互いにリンクされているか、又は、20に示すように階層方式でリンクされている。上述したように、この一次サーバ／二次サーバ・アーキテクチャは、実質的にあらゆる型のシステム（ピア・ツー・ピア・システムでも階層システムでも）との高信頼の同期を行うことができる。一次サーバ／二次サーバ・アーキテクチャを有するために、あらゆる型のシステムを適応ことができ、それによって、本システムは二次サーバ間のあらゆるトポロジに適応することができる。

【0081】図2において、一実施形態では、本システムのサーバのアーキテクチャは、多数のオブジェクトコンテナ22と、それに接続されたオブジェクトコンテナ・マネージャ24を含んでいる。オブジェクトコンテナ・マネージャ24は、シンクロナイザ・マネージャ26に接続されており、シンクロナイザ・マネージャ26は、オブジェクトコンテナ22及びシンクロナイザ28と接続されている。プロトコル・ユーティリティ30は、シンクロナイザ・マネージャ26によって駆動され、ネットワークに対し最も信頼性の高い接続を選択する。

【0082】動作時には、システム・ユーティリティ又はアプリケーションが、オブジェクトコンテナ22又はシンクロナイザ・マネージャ26のいずれかから同期を開始する。シンクロナイザ・マネージャ26は、プロトコル・ユーティリティ30に問い合わせ、同期させる2つのサーバ間に高信頼性接続をオープンさせる。その後、シンクロナイザ・マネージャ26は、プロトコル・ユーティリティ30からの結果に基づいて、シンクロナイザ28などのシンクロナイザを生成する。そして、2つのサーバ上のシンクロナイザ28は、同期化処理を開始する。

【0083】ここで、図3において、図2のシステムで利用されるサブルーチン又はメソッドについて説明する。オブジェクトコンテナ22は、putメソッド32、deleteメソッド34、getメソッド36、synchronizeメソッド38、applyUpdateメソッド40、generateUpdatesメソッド42、purgeOffHistoryメソッド44及びgetSummarizingVersionVectorメソッド46を利用する。これらのメソッドはJavaによって書かれており、後述するソース・コード中に見つけることができる。

【0084】図示しているように、オブジェクトコンテナ・マネージャ24には、openObjectContainerメソッド50が与えられており、シンクロナイザ28には、startメソッド52、nullメソッド54、nuchメソッド5

6及びstopメソッド58が与えられている。これらはすべてJavaで書かれており、後述するソース・コードに見つけることができる。

【0085】シンクロナイザ・マネージャ26には、synchronizerメソッド60が与えられており、プロトコル・ユーティリティ30には、標準のisConnectedメソッド62及びgetBestConnectionメソッド64が与えられている。

【0086】ここで、図4において、オブジェクトコンテナ22に使用される本システムのデータ構造を示す。重要なのは、この発明を通して使用される要約バージョンベクトル（ここでは70で示す）の構造である。図示しているように、オブジェクトコンテナ22の要約バージョンベクトルは、識別用フィールド72及びタイムスタンプ用フィールド74を有している。Sid1はシステムのオブジェクトコンテナ1を示し、1544という数は、オブジェクトコンテナ1に関連するタイムスタンプを示す。要約バージョンベクトルは、本質的に、個々のコンテナのオブジェクトすべてについての識別子及びタイムスタンプを含んでいる。リモート・オブジェクトコンテナについての同様の要約バージョンベクトルを、76で示す。このように、システム内の多数のロケーションに、要約バージョンベクトルがあることが分かる。

【0087】1セットのリモート・オブジェクトコンテナが、共通の差分更新を既に受信しているコンテナを予め定義しているか又は予め選択していることにより、80で示すように、オブジェクトコンテナから読取ることができる差分更新を識別する役割を果たす、最新共通先祖バージョンベクトルと呼ばれる特殊なバージョンベクトルがある。これによって、いくつかの更新が予め定義されたオブジェクトコンテナのセットに既に格納されているため、その更新をパージすることができる。

【0088】82で示すように、オブジェクトコンテナ22に対する更新の履歴を含むログ構造を示す。図示しているように、ログには、オブジェクト・バージョンベクトル86又は更新スタンプ88のいずれかを参照するフィールド84、及び対応するオブジェクト92又は差分更新94を参照するフィールド90が含まれる。更新94に關係するオブジェクト96には、追加の更新98が含まれており、それらはここで両方ともベース100で示す初期オブジェクトを更新したものである。

【0089】オブジェクト92は、常にオブジェクト全体として送信されなければならないオブジェクトであり、一方オブジェクト96は、ベースに対する差分更新と言えるオブジェクトである。

【0090】図5では、同期中にオブジェクトコンテナ内のデータを変更することができるシステムを示している。図示しているように、オブジェクトコンテナ1（ここでは、110で示す）には、112で示すサーバ1の要約バージョンベクトルVV1が与えられている。ま

た、オブジェクトコンテナ110には、サーバ1以外のサーバの要約バージョンベクトルVV1（ここでは114で示す）も与えられている。116で示すサーバ1の最新共通先祖バージョンベクトルCVV1もまた、オブジェクトコンテナ110に与えられている。同様に、118で示すサーバ1の更新ログ1もまた、オブジェクトコンテナ110に与えられている。

【0091】オブジェクトコンテナ110'は、指示された入力112'～118'を有するオブジェクトコンテナ110の動的に変化している状態を示している。オブジェクトコンテナ110の状態は、オブジェクトコンテナ120が、120'で示すように動的に変化し、矢印120で示すようにオブジェクトコンテナ110にデータを送信することによって動的に変化する。同期プロセスは、オブジェクトコンテナ110が、矢印122で示すように、オブジェクトコンテナ2（ここでは120で示す）に関連するサーバに対し要約バージョンベクトル1を送信することによって、開始する。オブジェクトコンテナ120は、要約バージョンベクトル1が供給する同期情報を含むように変化し、矢印124で示すように、情報を送信することによって、オブジェクトコンテナ110を更新する。

【0092】理解されるように、オブジェクトコンテナ120は、入力として、126で示すサーバ2の要約バージョンベクトルVV2、128で示すサーバ2以外のサーバの要約バージョンベクトルVV2、130で示すサーバ2の最新バージョンベクトルCVV2、及び132で示すサーバ2の更新ログ2を有する。

【0093】矢印122で示すように、送信される情報は、要約バージョンベクトル2に、オブジェクトコンテナ120内の差分同期をサポートするオブジェクトの識別子を足したものである。

【0094】矢印134で示すように、オブジェクトコンテナ110'からオブジェクトコンテナ120'へ情報が送られることによって、同期は継続する。矢印134で示すように転送される情報は、コンテナ1のオブジェクトがない差分同期をサポートするオブジェクトの識別子であるが、そのオブジェクトはオブジェクトコンテナ2内に存在する。これは、矢印124で示すようにオブジェクトコンテナ120'から受信したオブジェクトの識別子を、オブジェクトコンテナ1内のオブジェクトの識別子と比較することによって、確認することができる。なお、要約バージョンベクトルは、同期プロセスの第1段階で前もって送信されているため、この時点では送信されない。

【0095】矢印124に示すように情報が転送されることにより、オブジェクトコンテナ120'は、古い情報をパージすることができる。履歴上古い情報は、最新共通バージョンベクトルCVV2'（ここでは130'で示す）にアクセスするアトによって、パージされる。

【0096】その後、更新は、オブジェクトコンテナ120"からオブジェクトコンテナ110"に送信され、この時点で、パージ動作を行うことができる。すべてのパージ動作が行われた後、同期は完了し、オブジェクトコンテナ110"は、システムのすべてのオブジェクトの最終的に更新されたバージョンを含んでいる状態となる。オブジェクトコンテナ120内のオブジェクトもまた、同様に同期させ更新することができることがわかるだろう。

【0097】図6では、オブジェクトコンテナ内のオブジェクトを変更するシステムを示している。ここで、オブジェクトコンテナ140には、要約バージョンベクトル142及び更新ログ144が与えられている。146で示すような変更ステップでは、異なる要約バージョンベクトル（ここでは148で示す）と、それに関連する更新ログ（ここでは150で示す）を入力する必要がある。この図は、オブジェクトコンテナ140が変更された場合、変更を記録するために、対応する要約バージョンベクトルとその対応する更新ログとを変更する必要があることを示している。

【0098】ここで、図7では、同期プロトコルを示している。同期プロトコルには、2つのシンクロナイザが含まれており、1つは同期開始側であり、もう1つは同期応答側である。同期を開始するシンクロナイザは、まず、同期に回答するシンクロナイザが同期開始側に回答する用意ができた時、そのシンクロナイザに対し、プル／プッシュ（pull/push）要求を送信する。その後、この応答側のサーバは、要約バージョンベクトルを返信する。そして、次の要約バージョンベクトルが、差分同期をサポートするオブジェクトの識別子と共に反対方向に転送される。次に、オブジェクトの識別子が、そのまた反対方向に返送され、その結果、初期同期要求に回答するサーバに対し更新が送信される。

【0099】同期プロトコルの結果、同期を開始するサーバに最新共通先祖バージョンベクトルに対応する更新が与えられると共に、すべての差分更新がパージされる。同期要求に回答するサーバについても、同様である。

【0100】図8では、転送すべき更新を抽出する方法を示している。ここでは、サーバ160は、上述したフィールド72、74を含む関連する要約バージョンベクトル162を有している。強調されているフィールドは、第2のサーバ、すなわちサーバ164の要約バージョンベクトルにおける対応する値より時間的に早いタイムスタンプを示している。これは、サーバ164に関連する要約バージョンベクトル168の強調部166によって示す。

【0101】サーバ164の更新ログ170におけるバージョンベクトル又は更新スタンプをチェックすることにより、更新が抽出される。なお、これらバージョンベ

クトル又は更新スタンプは、図4において86で表すバージョンベクトルである。一実施形態において、チェック・プロセスは以下のように行う。すなわち、要約バージョンベクトル162において強調されている対応するタイムスタンプより大きく、かつ、要約バージョンベクトル168において強調されている対応するタイムスタンプより小さいか又はそれと等しい、少なくとも1つのタイムスタンプを含むバージョンベクトル又は更新スタンプを有するオブジェクト又は差分更新を、抽出する。

【0102】図9では、競合を検出し解消する方法を示し、ここでは、サーバ180が対応する要約バージョンベクトルを有している。更新ログ184は、サーバ180に接続されており、そのベース194の最上位に3つの差分更新188、190、192を含むオブジェクト186を有している。第2のサーバ、すなわちサーバ196には、要約バージョンベクトル198及び対応する更新ログ200が供給される。

【0103】サーバ196は、186との競合を示す異なる差分更新を有するという点を除いてはオブジェクト186と同じであるオブジェクト202を有する。これは、強調部204、206を強調部188と比較することによって示す。問題は、更新204、206が更新186と競合している理由をどのようにして見つけるかである。このような競合は、オブジェクト全体のバージョンベクトル又は更新スタンプ（この図では図示せず）、あるいはサーバ180内の差分更新188、190、192及びサーバ196内の差分更新204、206、208、210を、要約バージョンベクトル182、198の共通バージョンベクトル212と比較することによって、検出する。2つの要約バージョンベクトルの共通バージョンベクトルは、2つのサーバの共通する状態を反映している。共通バージョンベクトル212における対応するタイムスタンプより大きいタイムスタンプを有するサーバ180、196の双方において、オブジェクト186、202の差分更新がある場合、2つのオブジェクト186、202は競合している。

【0104】オブジェクト186、202が競合していることが見つかった後、予め決められた照合調整メソッドを呼出し、220で示すメソッドに競合している差分更新を渡すことにより、その競合が解消すなわち照合調整される。なお、照合調整メソッドは、通常、特定のアプリケーションに基づいて決定される。

【0105】図10では、更新ログをパージする1つの方法を示している。ここでは、サーバ228は、自身の要約バージョンベクトル（260で示す）、及び他のサーバからの要約バージョンベクトル262、260を有している。サーバ228の最新共通バージョンベクトル236は、3つのサーバすべての共通の状態を表す3つの要約バージョンベクトル230、232、234から計算される。

【0106】最新共通バージョンベクトル236を取得した後、更新ログ238内の対応する更新ログ・エントリをパージすることが可能となる。これは、括弧240でくくられたログ・エントリを削除又はパージすることができることを意味している。これは、これらの更新が、最新共通バージョンベクトル236より古い更新スタンプを有しているためであり、これらの更新が、上記3つのサーバすべて既に取込まれているということを反映している。

【0107】図11に示すように、オブジェクト全体の同期と差分同期とを切換えることができる。ここでは、オブジェクト全体の同期を250で示し、差分同期を252で示す。スイッチ254を用いて、どちらの同期プロセスを使用するかを判断する。スイッチ254は、以下のように動作するロジック256によって制御される。

【0108】一実施形態では、差分同期をサポートすることができるオブジェクトを、オブジェクト全体として送信することによって同期させるか、あるいはその差分更新の一部を送信することによって同期させるかを、以下のロジックで判断する。第1に、オブジェクトが第1のサーバにのみ存在し、第2のサーバにない場合、2つのサーバが互いに同期している時、オブジェクト全体を第1のサーバから第2のサーバへ送信する必要がある。

【0109】第2に、オブジェクトが2つのサーバに存在する場合であって、少なくとも1つのサーバにおいて、オブジェクトのベースの状態が2つのサーバにおけるオブジェクトの共通する状態より新しい場合、2つのサーバのどちらのオブジェクトも、他方のサーバから差分更新のみを受信した後、最新の状態で同期することができないため、2つのサーバの間でオブジェクトの差分同期を行うことができない。この場合、オブジェクト全体の同期に切換える必要がある。それ以外の場合は、差分更新を2つのサーバの間で実現することができ、システムを差分同期に切換えることができる。

【0110】オブジェクト全体の同期と差分同期との切替えのロジックに、必要であれば他の要素を加えることができる。例えば、差分同期からオブジェクト全体の同期への切替えを、メモリの空き領域がほとんど使い果たされている場合、又はその差分更新よりもオブジェクト全体を送信することがより効率的であると思われる場合に、トリガするようにしてもよい。あるいは、オブジェクト全体の同期から差分同期への切替えにより、転送されるデータを最小にすることができる。言換えれば、オブジェクト全体と比べてオブジェクトの差分更新を送信の方が効率的である場合、差分同期に切換える方が望ましい。なお、オブジェクト全体の同期と差分同期との間の切替えは、個々のオブジェクトに基づいて決定される。

【0111】特に、本システムで用いられるデータは

ファイル、文書、データベース又はこれらの一部あるいはすべての混合であってよく、それらは適当に加工される。データをグループ化する方法は、アプリケーション毎に異なる。例えば、ファイル・システムでは、ディレクトリをファイル・グループとしてみなすことができ、データ（ファイル）を編成するために内部の構造をさらに定義することができる。

【0112】オブジェクト指向データベースでは、データ・グループに、そのアプリケーションをサポートする必要がある同質又は異質のオブジェクトすべてを含むことができる。第1のステップとして、互いに同期させる必要がある異なる装置のデータ・グループすべてが、結局まったく同じデータ・セットを含むものとする。

【0113】本システムは、Javaを使用して実現されている。従って、以下の説明は、Javaの専門用語に基づいている。オブジェクトコンテナは、Javaのオブジェクト・グループを格納及び保持するコンテナである。オブジェクトコンテナ内のオブジェクトは、replace/applyメソッド及びreconcileメソッドを供給するJavaの「直列化可能オブジェクト」である同期可能オブジェクトによって表される。

【0114】同期可能オブジェクトは、ファイル、文書又はデータベース・テーブルの行のような具体的なデータを抽象化したものである。（同期可能）オブジェクトの全セットが、あるオブジェクトコンテナ内にすでに存在すると仮定すると、アプリケーションは、そのget又はputメソッドを呼出すことによって、オブジェクトコンテナ内のオブジェクトにアクセスすることができる。

【0115】サーバにおいて、オブジェクトコンテナ・マネージャのopenObjectContainerメソッドを呼出すことにより、複数のオブジェクトコンテナを生成することができる。各サーバでは、1つのオブジェクトコンテナ・マネージャしか存在することができず、それを用いてサーバでのオブジェクトコンテナすべてを管理する。オブジェクトコンテナを生成するだけでなく、生成されたオブジェクトコンテナをリスト化するか又は削除するといったサービスもまた提供する。便宜上、各オブジェクトコンテナは、生成された時に、全体的に一意的識別子が割当てられ、その識別子は、本システムのそのオブジェクトコンテナを参照する時に使用することができる。一実施形態では、オブジェクトコンテナの識別子は、入力としてオブジェクトコンテナのURLを用いて一方ハッシュ関数によって生成されるハッシュ・コードに割当てることができる。

【0116】2つのオブジェクトコンテナを一貫した状態にするために、シンクロナイザは、本システムの重要な要素である。各サーバに、異なる型の通信トランスポート及びプロトコルを使用する複数のシンクロナイザを設けることができる。例えば、シンクロナイザ1は、TCP/IP及び両方向同期プロトコルを使用し、シンク



ロナイザ2は、赤外線プロトコル及び信頼性の低い無線通信により適した一方向プロトコルを使用するようにしてもよい。シンクロナイザは、必要な時にシンクロナイザ・マネージャによって動的に生成され、同期を終了した直後に自己消失する。

【0117】サーバ内のシンクロナイザ・マネージャは、シンクロナイザと同期する予定であるオブジェクトコンテナと結合してシンクロナイザが同期を開始するのを止める(trig)ようにしてもよいし、又は、リモート・サーバのシンクロナイザ・マネージャからの1又は複数の同期要求に従う(listen)ようにしてもよい。プロトコル・ユーティリティは、シンクロナイザ・マネージャが、現在使用可能な最良の接続に合致する生成すべきシンクロナイザの適切な選択に役立つ。プロトコル・ユーティリティは、Java拡張パッケージが供給するクラスの1つである。

【0118】上述した説明から分かるように、オブジェクトの内容を表すために使用されるフォーマット、及び平行な更新競合を照合調整するために使用されるアルゴリズムのような、オブジェクトのセマンティクスが、シンクロナイザから切離されている。このような特徴により、本システムが、異なるアプリケーション・システム間で同期することができるようになる。

【0119】サーバ1が、そのオブジェクトコンテナ1をサーバ2のオブジェクトコンテナ2と同期させるよう指定されているとする。また、オブジェクトコンテナ1(i=1、2)にバンドルされた要素は、すべてシンクロナイザ・マネージャ1、シンクロナイザ1等のように注釈を付けるものとする。同期プロセス全体の詳細なステップは、以下の通りである。

【0120】シンクロナイザ・マネージャ1は、通信チャンネルをオープンにし、同期要求に従っているシンクロナイザ・マネージャ2に接続する。シンクロナイザ・マネージャ1は、同期すべきオブジェクトコンテナ及び同期に使用するシンクロナイザを示す同期要求を、シンクロナイザ・マネージャ2に送信する。

【0121】シンクロナイザ・マネージャ1、2は、一組の同じシンクロナイザ1、2を作成して接続する。ここで、シンクロナイザ1は同期開始側とし、シンクロナイザ2は応答側とする。

【0122】シンクロナイザ1は、シンクロナイザ2に対しpull又はpush更新コマンドを送信する。pull更新コマンドは、オブジェクトコンテナ1がオブジェクトコンテナ2からの更新を受信する必要があることを意味しており、push更新コマンドは、オブジェクトコンテナ1がオブジェクトコンテナ2に対して更新を伝搬することを意味している。シンクロナイザ1がpull又はpush更新コマンドのいずれを送信すべきか、あるいは順次両方を送信すべきかは、同期を開始するシステム・ユーティリティマはアプリケーションによって判断される。

【0123】以下のステップでは、シンクロナイザ1がシンクロナイザ2にpush更新コマンドを送信したと仮定する。シンクロナイザ1がシンクロナイザ2にpull更新コマンドを送信した場合、シンクロナイザ2は、シンクロナイザ1に対して確認メッセージを返送し、残りのステップは上記ステップのまったく逆(鏡像)となる。シンクロナイザ1は、オブジェクトコンテナ1の図3に示すようなgetSummarizingVersionVectorメソッドを呼出すことによって、オブジェクトコンテナ1の要約バージョンベクトルを取得し、シンクロナイザ2に対しその要約バージョンベクトルを送信する。オブジェクトコンテナ1の要約バージョンベクトルは、オブジェクトコンテナ1の現在の状態を反映している。

【0124】シンクロナイザ2は、必要な時、受信したオブジェクトコンテナ1の要約バージョンベクトルを、オブジェクトコンテナ2内に記録する。そして、シンクロナイザ2は、図3に示すgetSummarizingVersionVectorメソッドを呼出すと共に、後述するソース・コードに示すオブジェクトコンテナ2のentriesメソッドを呼出すことにより、オブジェクトコンテナ2の要約バージョンベクトルと、オブジェクトコンテナ2内の、差分同期をサポートするオブジェクトの識別子とを取得する。オブジェクトコンテナ2の要約バージョンベクトルは、オブジェクトコンテナ2の現在の状態を反映している。そして、シンクロナイザ2は、上記のように取得した要約バージョンベクトル及び識別子をシンクロナイザ1に送信する。

【0125】シンクロナイザ1は、必要な時、受信したオブジェクトコンテナ2の要約バージョンベクトルを、オブジェクトコンテナ1内に記録する。そして、シンクロナイザ1は、受信したオブジェクトコンテナ2の要約バージョンベクトルと受信した識別子とをアーギュメントとして、後述するソース・コードに示すオブジェクトコンテナ1のgetWholeObjectIdsメソッドを呼出すことにより、オブジェクト全体として受信しなければならないオブジェクトの識別子を取得する。

【0126】上記のように取得された識別子には、オブジェクトコンテナ2には存在するがオブジェクトコンテナ1には無いオブジェクト、及びベースの状態がオブジェクトコンテナ1及びオブジェクトコンテナ2の共通の状態より新しいオブジェクトの識別子が含まれている。その後、シンクロナイザ1は、上記で取得した識別子をシンクロナイザ2に送信する。

【0127】シンクロナイザ2は、受信したオブジェクトの識別子とステップ6で受信したオブジェクトコンテナ1の要約バージョンベクトルとをアーギュメントとして、オブジェクトコンテナ2の図3に示すようなgenerateUpdatesメソッドを呼出すことにより、オブジェクトコンテナ1に送信しなければならない更新を取得する。上記のように取得した更新の各々はバージョンベクトル

ルを有するオブジェクト全体でも、又は更新スタンプを有する差分更新でもよい。そして、シンクロナイザ2は、上記のように取得した更新を1つずつシンクロナイザ1に送信する。シンクロナイザ1は、オブジェクトコンテナ1の図3に示すようなapplyUpdateメソッドを呼出すことにより、受信した更新の各々オブジェクトコンテナ1に与える。

【0128】更新を送信又は受信した後、シンクロナイザ1、2は、オブジェクトコンテナ1又はオブジェクトコンテナ2に関係があるすべてのオブジェクトコンテナに含まれている差分更新をパージするために、オブジェクトコンテナ1及びオブジェクトコンテナ2の図3に示すpurgeOffHistoryメソッドを呼出すことができる。

【0129】オブジェクトコンテナのapplyUpdateメソッドは、オブジェクトを、受信したオブジェクトに置換えるか、オブジェクトを、受信したオブジェクトと照合調整するか、オブジェクトに受信した差分更新を加えるか、又は受信した差分更新をそれと競合するオブジェクトの差分更新と照合調整するか、いずれかのために、オブジェクト内で定義されたメソッドの1つをさらに呼出す。また、それは、オブジェクトコンテナが既にその中に更新を配置したという事実を表すために、オブジェクトコンテナの要約バージョンベクトルを更新する。

【0130】上述した説明から分かるように、本システムで展開する要約バージョンベクトルは、並行な更新競合の検出にのみ使用するのではなく、更新ログの中のエントリ数を減らすためにも使用する。

【0131】オブジェクトコンテナの要約バージョンベクトルは、実質的に、オブジェクトコンテナ内のすべてのオブジェクトのバージョンベクトル及びすべての差分更新の更新スタンプを要約したものである。差分同期をサポートすることができないオブジェクトのバージョンベクトルは、オブジェクト全体の現在の状態を反映し、差分同期をサポートすることができるオブジェクトのバージョンベクトルは、関連する差分更新がすべて無いオブジェクトのベースの現在の状態を反映する。オブジェクト全体についてのバージョンベクトルのように、オブジェクトのベースのバージョンベクトルもまた、関連する差分更新の一部又は全部がパージされるのに従って動的に変化する。従って、オブジェクトのバージョンベクトルは、オブジェクト全体又はオブジェクトのベースの現在の状態を特徴付ける。

【0132】オブジェクトコンテナの要約バージョンベクトル又はオブジェクトのバージョンベクトルは、更新スタンプのベクトルである。更新スタンプは、2つのフィールドからなる。すなわち、第1のフィールドは、ベクトルがオブジェクトのバージョンベクトルである場合はオブジェクトに対し、又はベクトルがオブジェクトコンテナの要約バージョンベクトルである場合はオブジェクトコンテナ内のオブジェクトに対し、変更を行ったオ

ブジェクトコンテナの識別子である。

【0133】また、第2のフィールドは、オブジェクトに対し最後の変更を行った時、第1のフィールドに対応する変更するオブジェクトコンテナを変更することによって生成されたタイムスタンプである。便宜上、更新スタンプは(sidi、ti)で表す。ここで、sidiは更新スタンプの第1のフィールドであり、tiは更新スタンプの第2のフィールドである。更新スタンプ内のタイムスタンプは、実時間、更新一連番号、又はアプリケーションにとって意義深いものにすることができる。

【0134】オブジェクトコンテナが生成する一連のタイムスタンプは、意味が一貫しており、まったく単調でなければならないが、異なるオブジェクトコンテナは、それらが生成するタイムスタンプを有する異なるセマンティクスを有することができる。これは、異なるアプリケーション・システムが同じセマンティクスでタイムスタンプを生成することを必要とするのは実際的でないため、異なるアプリケーション・システムに互ってオブジェクトコンテナ間の同期をとる場合に、非常に重要である。

【0135】上述した更新スタンプの表現を用いて、オブジェクトコンテナの要約バージョンベクトル又はオブジェクトのバージョンベクトルを、{(sidi、ti)、(sidi、ti)、(sidi、ti)}又は{(sidi、ti)|i=1、2、n}と表すことができる。オブジェクトコンテナの要約バージョンベクトル及びオブジェクトのバージョンベクトルのセマンティクスは、オブジェクトのバージョンベクトルが、それを包含しているオブジェクトコンテナの現在の状態ではなく、包含しているオブジェクトコンテナ内のオブジェクトの現在の状態のみを反映しているという点を除けば、同様である。これらが同様であるため、要約バージョンベクトルについてのみ以下に詳細を説明する。オブジェクトのバージョンベクトルは、必要な時にのみ言及する。

【0136】オブジェクトコンテナの要約バージョンベクトルのi番目の要素は、オブジェクトコンテナが、タイムスタンプtiが示す時間まで識別子sidiを有するオブジェクトコンテナが行う変更をすべて含んでいることを意味する。便宜上、オブジェクトコンテナjの要約バージョンベクトルをvviと表し、そのi番目の更新スタンプをvviと表す。

【0137】オブジェクトコンテナjの要約バージョンベクトルvviのサイズnは、j番目のオブジェクトコンテナが、その中のオブジェクトに対して、タイムスタンプt1、t2、tnまで識別子sidi、sid2、sidnを有するオブジェクトコンテナが行う変更をそれぞれすべて含んでいることを意味する。上述した説明に基づき、要約バージョンベクトル、バージョンベクトル及び更新スタンプを利用して、2つのエンティティの状態を比較することができるように、以下にいくつかの関数/命題を

定義/証明する。以下、エンティティは、オブジェクトコンテナ、オブジェクト又は差分更新を表す。

【0138】定義： $z=y.getTime(x)$ は、識別子 $x$ を有するオブジェクトコンテナに対応する $y$ のタイムスタンプを戻す関数である。 $y$ は、オブジェクトコンテナの要約バージョンベクトル、オブジェクトのバージョンベクトル又は差分更新の更新スタンプを表している。 $x$ が $y$ で表されない場合、戻される更新スタンプ $z$ は0である。

【0139】定義： $y_1$ 及び $y_2$ を、オブジェクトコンテナの要約バージョンベクトル、オブジェクトのバージョンベクトル又は差分更新の更新スタンプのいずれかとする。 $y_1$ 及び $y_2$ 内のオブジェクトコンテナの識別子について $y_1.getTime(x)=y_2.getTime(x)$ である場合、 $y_1$ は、 $y_2$ と等しい。 $y_1$ 及び $y_2$ 内のオブジェクトコンテナの識別子について $y_1$ が $y_2$ と等しくなく、かつ $y_1.getTime(x)>y_2.getTime(x)$ である場合、 $y_1$ は、 $y_2$ より新しい。 $y_1$ 及び $y_2$ 内のオブジェクトコンテナの識別子について $y_1$ が $y_2$ と等しくなく、かつ $y_1.getTime(x)<y_2.getTime(x)$ である場合、 $y_1$ は、 $y_2$ より古い。 $y_1$ が $y_2$ と等しくなく、 $y_2$ より新しくなく、かつ $y_2$ より古くない場合、 $y_1$ は $y_2$ と競合する。

【0140】命題： $e_1$ 及び $e_2$ は、オブジェクトコンテナ、オブジェクト、オブジェクトのベース又は差分更新のいずれかを表すものとし、 $y_1$ 及び $y_2$ は、 $e_1$ 及び $e_2$ それぞれの要約バージョンベクトル、バージョンベクトル又は更新スタンプであるものとする。 $y_1$ が $y_2$ と等しい場合、 $e_1$ の現在の状態は、 $e_2$ と同じか又は一致する。 $y_1$ が $y_2$ より新しい場合、 $e_1$ の現在の状態は $e_2$ より新しい。 $y_1$ が $y_2$ より古い場合、 $e_1$ の現在の状態は、 $e_2$ より古い。 $y_1$ が $y_2$ と競合する場合、 $e_1$ の現在の状態は、 $e_2$ と競合する。

【0141】証明：定義により、 $y_1$ が $y_2$ と等しい場合、 $e_1$ に含まれる更新はすべて $e_2$ にも含まれ、その逆でもある。明らかに、 $y_1$ が $y_2$ と等しい場合、 $e_1$ の現在の状態は $e_2$ と同じである。次に、 $y_1$ が $y_2$ より新しく、 $e_1$ の現在の状態が $e_2$ より新しくないものとする。定義により、 $y_1$ が $y_2$ より新しい場合、 $e_1$ は、 $e_2$ に含まれていない更新を含んでいなければならない。そのため、上記仮定は妥当ではない。命題の他の部分も同様に証明することができる。（結果を証明。）

【0142】定義： $y_1$ 及び $y_2$ は、それぞれオブジェクトコンテナ $e_1$ 及び $e_2$ の要約バージョンベクトルであるものとする。 $r_y$ が、 $y_1$ 及び $y_2$ の両方に存在するオブジェクトコンテナの識別子のみを含み、そのような識別子の各々に対応する $r_y$ 内のタイムスタンプが、 $y_1$ 及び $y_2$ 内の対応するタイムスタンプの小さい方である場合、 $y_1$ 及び $y_2$ の共通のバージョンベクトルを $r_y$ と呼ぶ。

【0143】命題： $e_1$ 及び $e_2$ は、2つの異なるオブジェクトコンテナであるものとし、 $v_1$ 及び $v_2$ はそれぞれ

及び $y_2$ の要約バージョンベクトルであるものとする。さらに、 $r_y$ は、 $y_1$ 及び $y_2$ の共通のバージョンベクトルであるものとする。従って、 $r_y$ は、 $e_1$ 及び $e_2$ がそこから分かれた状態を表す。

【0144】証明： $y_1$ 及び $y_2$ の定義により、更新が $e_1$ 及び $e_2$ の両方に含まれている場合、それが $y_1$ 及び $y_2$ の両方に反映されていなければならない。そのため、更新に関するタイムスタンプは、 $y_1$ 及び $y_2$ 内の対応するタイムスタンプと等しいかそれより小さくなければならない。言換えれば、更新に関するタイムスタンプは、 $y_1$ 及び $y_2$ 内の2つの対応するタイムスタンプの小さい方と等しいかそれより小さい。共通のバージョンベクトルの定義により、更新は $r_y$ に反映される。

【0145】更新が $e_1$ 又は $e_2$ のいずれかにのみ含まれる場合、更新に関するタイムスタンプは、他方のオブジェクトコンテナの要約バージョンベクトルにおける対応するタイムスタンプより大きくななければならない。従って、更新は、更新の無いオブジェクトコンテナの要約バージョンベクトルによって反映され得ない。また、共通のバージョンベクトルの定義により、更新は、 $r_y$ によっても影響され得ない。（結果を証明。）

【0146】オブジェクトコンテナに対するオブジェクトの変更を行う度に、オブジェクトコンテナの図3に示すputメソッドを呼出すことにより、その変更はオブジェクトコンテナに反映されることが知られている。そのputメソッドにより、オブジェクトコンテナは、その要約バージョンベクトルを更新するか、変更すべきオブジェクトのバージョンベクトルを更新するか、又は加えるべき差分更新に関する更新スタンプを生成し、変更したオブジェクト又は加えた差分更新に対する参照、及び対応するバージョンベクトル又は更新スタンプに対する参照を、オブジェクトコンテナの更新ログに加える。差分更新の更新スタンプは、差分更新が、更新スタンプ内の第2のタイムスタンプに示す時刻に、更新スタンプの第1のフィールドに示す識別子で、オブジェクトコンテナによって生成されたことを意味する。

【0147】上述した説明に基づき、あるオブジェクトコンテナの他のオブジェクトコンテナとの差を理解し、2つのオブジェクトコンテナ間の並行な更新競合を検出し、更新ログ内のエントリの数を減少させることが可能である。 $l_s$ が、 $R_s$ で示す他のオブジェクトコンテナ $j$ から更新を取出す(pull)オブジェクトコンテナ $i$ を示すものとする。上述した計算による具体的なアルゴリズムは、以下の通りである。

【0148】差分計算 要約バージョンベクトル $svv^i$ を有する $R_s$ が、 $l_s$ の要約バージョンベクトル $svv^i$ と、オブジェクト全体として $l_s$ に送信する必要のあるオブジェクトの識別子 $E^i$ とを受信したものとする。そして、 $svv^i$ 及び $svv^j$ の共通のバージョンベクトル $rvv^{ij}$ は、上記定義に基づいて計算する事ができ、 $l_s$ に送信する必要のあ

る更新はすべて以下のように決定される。

【0149】第1に、差分同期をサポートしないオブジェクトについて、オブジェクトのバージョンベクトルをチェックする。そのバージョンベクトルが $rvv_i$ より新しい場合、オブジェクト全体が $ls$ に対して更新として送信される。U1を、上記計算によって得られる更新のグループとする。

【0150】第2に、差分同期をサポートするオブジェクトについて、オブジェクトのベースのバージョンベクトルをチェックする。そのベースのバージョンベクトルが $rvv_i$ より新しい場合、オブジェクト全体が $ls$ に対して更新として送信される。U2を、上記計算によって得られる更新のグループとする。

【0151】次に、差分同期をサポートするオブジェクトについて、それに差分更新が与えられている場合、その差分更新の各々の更新スタンプをチェックする。オブジェクトのそのような差分更新の更新スタンプのうち、 $rvv_i$ より新しいか競合しているものがあり、そのオブジェクトの識別子が $E_i$ に含まれている場合、そのオブジェクト全体が $ls$ に更新として送信される。あるいは、差分更新の更新スタンプが $rvv_i$ より新しいか競合している場合、オブジェクトの差分更新が $ls$ に対して更新として送信される。U3を、上記計算によって得られる更新のグループとする。

【0152】最後に、Uを、U1、U2及びU3の集合体とする。Uにおけるすべての更新の順序は、ある更新が、そのバージョンベクトル又は更新スタンプが他の更新のバージョンベクトル又は更新スタンプより新しい場合、その第1の更新は第2の更新より前にくる、といった順序である。Uの中の2つの更新が競合するバージョンベクトル又は更新スタンプを有する場合、2つの更新が差分更新であるならば、2つの更新の間の順序は、 $Rs$ の更新ログにおける順序と一貫していなければならない、そうでなければ、 $v1$ 及び $v2$ をそれぞれ2つの更新のバージョンベクトル又は更新スタンプとした場合、2つの更新は以下のように順序付けられる。

【0153】続いて、最初から最後まで $v1$ 内のオブジェクトコンテナの識別子 $sli$ を取出して、条件の1つを満たす $sli$ が見つかるまで以下に示す条件を調べる。 $v1.getTime(sli) > v2.getTime(sli)$ のような $sli$ が存在する場合、 $v2$ を有する更新の前に $v1$ を有する更新を配置する(put)。 $v1.getTime(sli) < v2.getTime(sli)$ のような $sli$ が存在する場合、 $v2$ を有する更新の後で $v1$ を有する更新を配置する。同じオブジェクトコンテナが生成したタイムスタンプの特徴が単調であるため、同じオブジェクトコンテナ内に同じバージョンベクトルは2つないことから、最終的に条件の1つが満たされる。

【0154】そして、Uは、 $ls$ に送信される、 $ls$ の $Rs$ との違いを表す更新のセットである。並行競合の検出  $ls$ 及び $R_r$ にそれぞれ含まれる2つのオブジェクト間の並行な

更新競合は、 $ls$ が2つのオブジェクトに関する更新を受信した後に検出される。

【0155】受信した更新が、そのバージョンベクトルと共にオブジェクト全体である場合、受信した更新のバージョンベクトルは、 $ls$ 内のオブジェクトのバージョンベクトルと比較される。受信したオブジェクトが $ls$ 内のオブジェクトのバージョンベクトルより新しいか又は古いバージョンベクトルを有する場合、2つのオブジェクト間に競合はなく、受信したオブジェクトは、 $ls$ 内のオブジェクトを置換えるか、又は捨てられる。2つのオブジェクトのバージョンベクトルが競合する場合、2つのオブジェクトは競合しており、 $ls$ 及び $R_r$ 内のオブジェクトの予め定義されたreconcileメソッドが呼出される。

【0156】受信した更新がその更新スタンプを有する差分更新である場合、 $ls$ 内の対応するオブジェクトに関する差分更新は、 $l$ 及び $R$ の要約バージョンベクトルの共通するバージョンベクトル $rvv_i$ と比較される。 $ls$ 内のオブジェクトの差分更新に、 $rvv_i$ より新しいか又はそれと競合しているものが無い場合、 $ls$ 内のオブジェクトと $R_r$ 内の対応するオブジェクトの間に競合は無い。そうでなければ、 $rvv_i$ より新しいか又はそれと競合する $ls$ 内のオブジェクトの差分更新はすべて、受信した更新と競合し、その競合を解消するために $ls$ 内のオブジェクトのreconcileメソッドに渡される。

【0157】更新ログ削減 第1のオブジェクトコンテナを呼出した $ls$ 及び $Rs$ の各々について、第2のオブジェクトコンテナを呼出した他方のオブジェクトコンテナが、第1のオブジェクトコンテナの更新ログ削減のプロセスにおいて考慮される第1のオブジェクトコンテナが選択したオブジェクトコンテナのセットの中にある場合、要約オブジェクトコンテナの要約バージョンベクトルは、第1のオブジェクトコンテナ内に記録され、更新ログ削減の以下のステップが実行される。

【0158】まず、第1のオブジェクトコンテナに記録された他のオブジェクトコンテナの要約バージョンベクトルがチェックされる。第1のオブジェクトコンテナが選択したすべてのオブジェクトコンテナの要約バージョンベクトルが記録されると、第1のオブジェクトコンテナの最新共通バージョンベクトルが計算されるか又は再計算される。ここで計算された第1のオブジェクトコンテナの最新共通バージョンベクトルには、第1のオブジェクトコンテナに記録された他のオブジェクトコンテナの要約バージョンベクトル及び第1のオブジェクトコンテナ自身の要約バージョンベクトルの各々に含まれるオブジェクトコンテナ識別子のみが含まれる。

【0159】第1のサーバの最新共通バージョンベクトルにおける各タイムスタンプは、第1のオブジェクトコンテナに記録された他のオブジェクトコンテナの要約バージョンベクトル、及び第1のオブジェクトコンテナ自身の要約バージョンベクトルにおける対応するタイムス

タンプの最も小さいものである。次に、第1のオブジェクトコンテナの最新共通バージョンベクトルが更新される場合、第1のオブジェクトコンテナの更新ログ内のすべてのログ・エントリがスキャンされ、更新、すなわち、オブジェクト全体、及びバージョンベクトル又は更新スタンプが第1のオブジェクトコンテナの最新共通バージョンベクトルより古い又は等しい差分更新の記録の削除部分が、第1のオブジェクトコンテナの更新ログからパージされる。

【0160】 上述した定義から分かるように、第1のオブジェクトコンテナの最新共通バージョンベクトルは、第1のオブジェクトコンテナが選択した他のオブジェクトコンテナすべてと第1のオブジェクトコンテナ自身とに共通するオブジェクトコンテナの最新状態を表す。従って、第1のオブジェクトコンテナの更新ログで更新をパージするセマンティクスは、第1のオブジェクトコンテナが選択した他のオブジェクトコンテナすべてに伝搬された更新が、第1のオブジェクトコンテナから結果的に削除される、ということである。これにより、概して、第1のオブジェクトコンテナが使用するメモリ及びディスク領域の一部を解放することができる。

【0161】 ここでは、この発明のいくつかの実施の形態について説明してきたが、当業者にとって、上述した内容が、単なる例として提示されているものであって、単に例示的であり限定しているのではないことは、明らかである。多数の変更及び他の実施の形態が、当業者の範囲内にあり、この発明と均等な特許請求の範囲によって定義される発明の範囲内にあるよう企図される。

#### 【0162】

【発明の効果】 以上のように、この発明によれば、広域モバイルコンピューティングに適応し、サーバを同期させ、同時にプロセスをより効率的にするための汎用システムを提供することができる。

【0163】 また、二次サーバが概して信頼性の低いサーバによりリンクされた同期プロセスのバックボーンを形成する、一次サーバの高性能高信頼性リンクを用いたネットワークを備えることで、モバイルコンピュータからの同期を、クライアント／サーバ・モード及びピア・ツー・ピアの双方で行うことができ、二次サーバのいかなるトポロジをもサポートすることができる。

【0164】 また、一次サーバは自動的に、かつ頻繁に同期をとるが、二次サーバの同期は、意図しない同期を避けるユーザの制御下で行われ、要約バージョンベクトルを使用して、オブジェクト毎にバージョンベクトルを交換する必要を無くすことにより、転送されるデータの量を最小にすることができる。

【0165】 また、この要約バージョンベクトルにより、要約バージョンベクトル及び更新スタンプを使用する差分同期、サーバ上の差分更新をパージするための最新共通バージョンベクトルの生成 影響を受けていない

サーバからのデータを用いた前の失敗地点からの同期の再開、及び転送されるデータのATOMとして差分更新を可能にすることによるきめの細かい同期が可能になる。

【0166】 さらに、オブジェクト全体の同期と差分同期とを自動的に切替える。また、本システムでは、更新が標準フォーマットで抽出され標準プロトコルに基づいて同期するため、データのセマンティクスが同期化から切離されていることから、異なるシステム間の同期が可能となる。

#### 【図面の簡単な説明】

【図1】 一次サーバ／二次サーバ構造を説明するこの発明のシステム・アーキテクチャを表す図である。

【図2】 オブジェクトコンテナ・マネージャ、同期マネージャ、シンクロナイザ及びオブジェクトコンテナを示す、図1のシステムのサーバのアーキテクチャを示す図である。

【図3】 図2に示す各構成要素についての主なサブルーチンを示す図である。

【図4】 図2のオブジェクトコンテナの構造を示すブロック図であり、その中で用いられる要約バージョンベクトル、最新共通先祖バージョンベクトル及び本システム内で用いられる差分ベースのログの構造を示す図である。

【図5】 要約バージョンベクトルの利用を説明する2つのサーバの同期を示す図である。

【図6】 オブジェクトコンテナ内でオブジェクトを変更するプロセスを示すブロック図である。

【図7】 第1及び第2のサーバ間でのデータの交換を示す同期プロトコルを示す図である。

【図8】 第2のサーバについての要約バージョンベクトル及び差分更新ログを利用して、第1のサーバから第2のサーバへ転送される更新を抽出するシステムを示すブロック図である。

【図9】 2つのサーバ間の競合を検出し解消するプロセスを示すブロック図である。

【図10】 サーバにおいて更新ログをパージするプロセスを示すブロック図である。

【図11】 システムの予め決められた特徴に基づき、オブジェクト全体の同期プロセスから差分同期プロセスへ同期システムを切替えることを説明するブロック図である。

#### 【符号の説明】

10 ネットワーク、12 一次サーバ、14 高性能リンク、16 二次サーバ、22 オブジェクトコンテナ、24 オブジェクトコンテナ・マネージャ、26 シンクロナイザ・マネージャ、28 シンクロナイザ、30 プロトコル・ユーティリティ、70 要約バージョンベクトル、72 識別用フィールド、74 タイムスタンプ用フィールド、80 最新共通先祖バージョンベクトル 82 ログ 92 オブジェクト 94 差

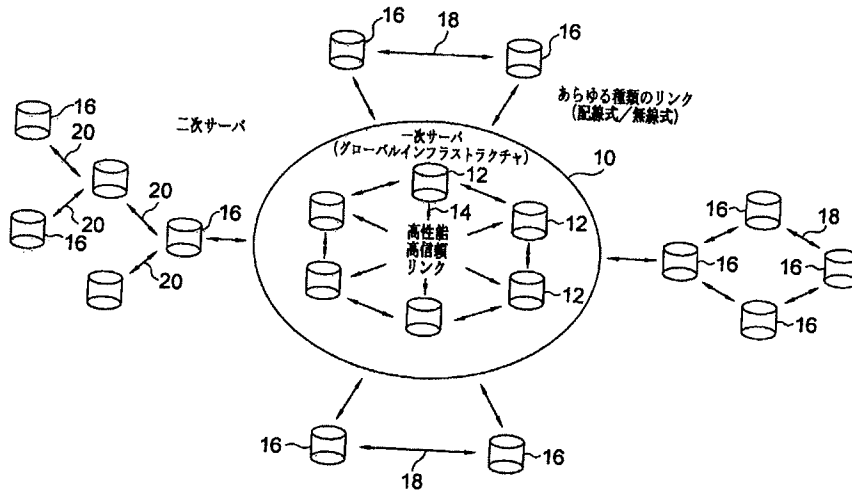
39

分更新、110 オブジェクトコンテナ1、120 オブジェクトコンテナ2、140 オブジェクトコンテナ、160 サーバ1、164 サーバ2、250 オ

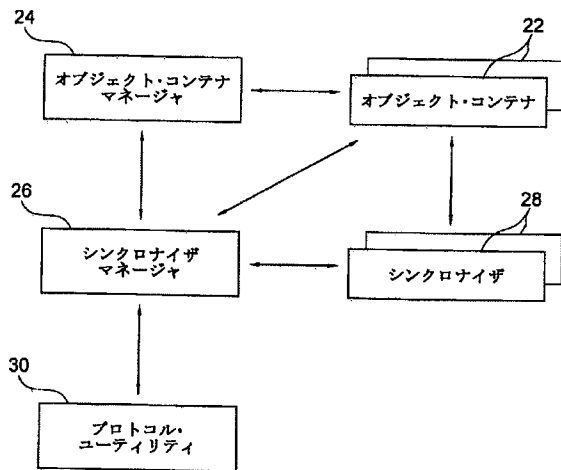
40

ブジェクト全体の同期、252 差分同期、254 スイッチ、256 ロジック。

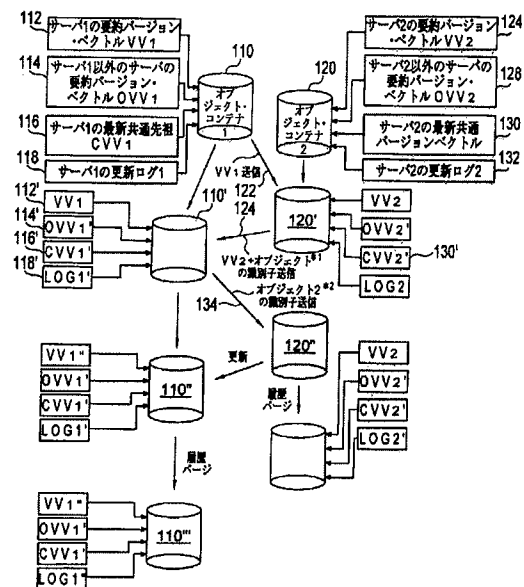
【図1】



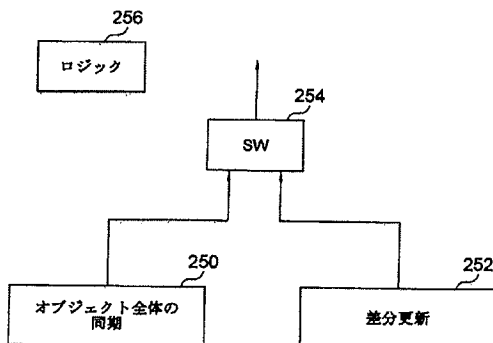
【図2】



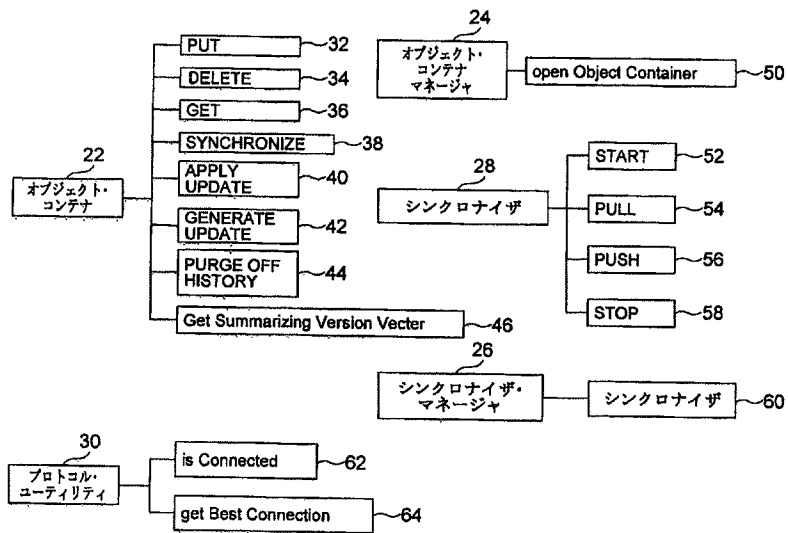
【図5】



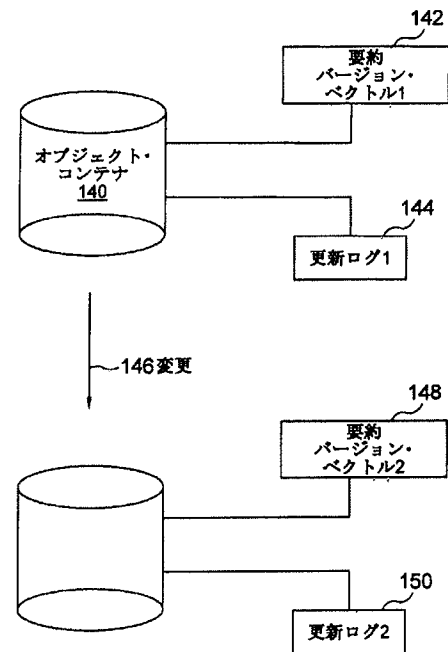
【図11】



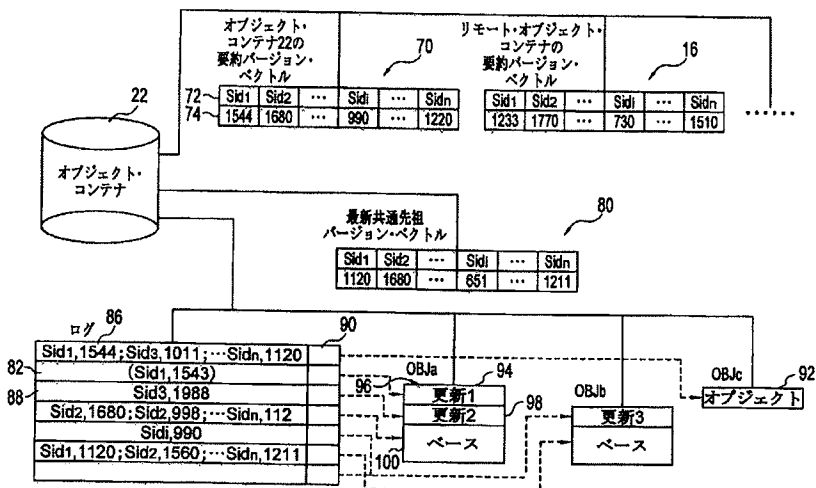
【図3】



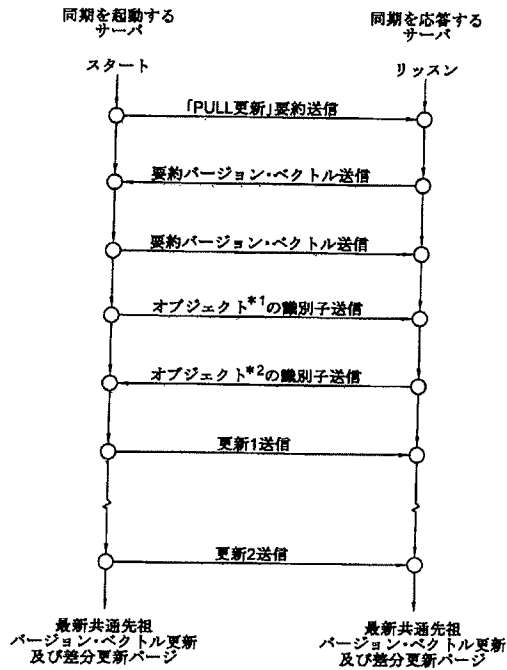
【図6】



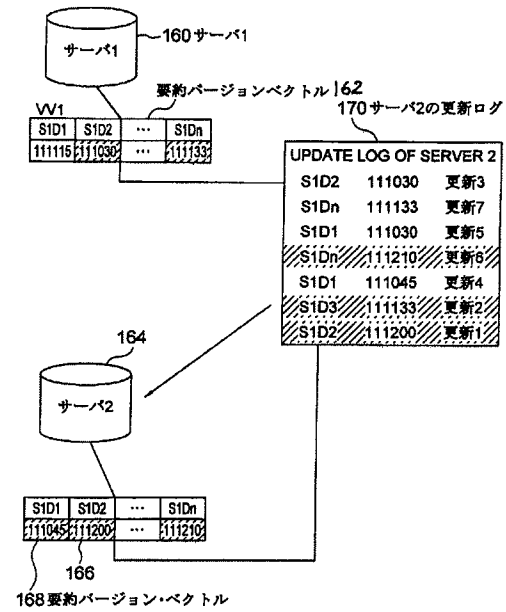
【図4】



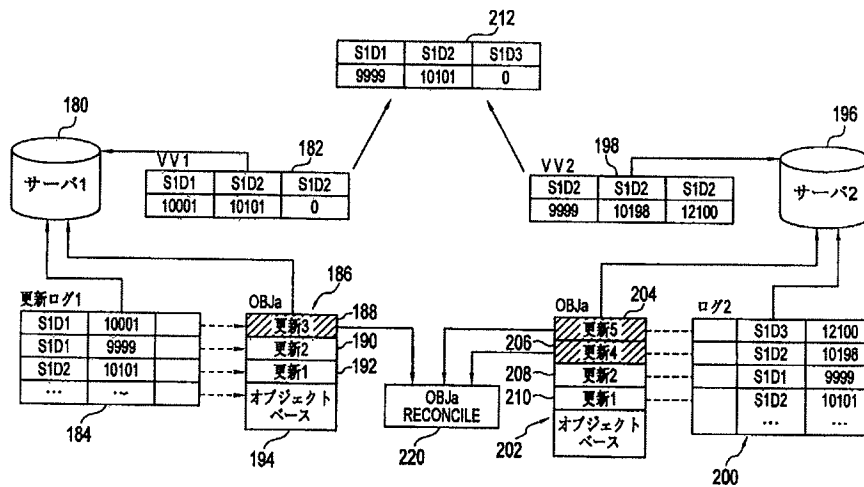
【図7】



【図8】

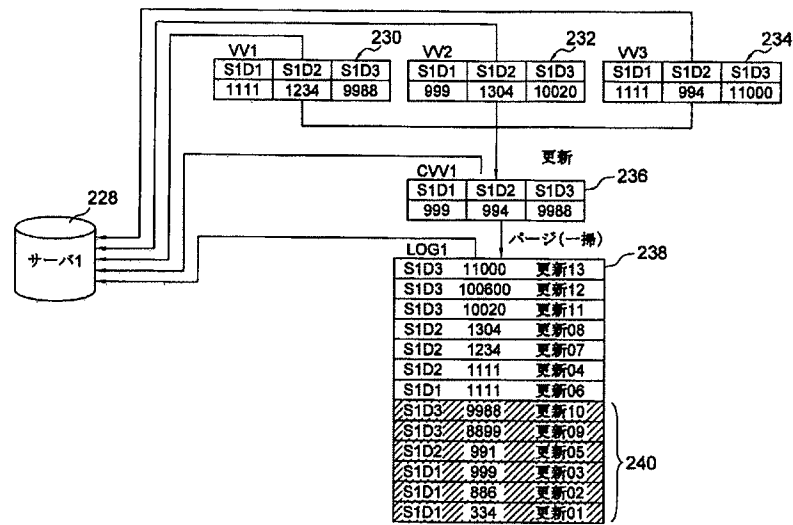


【図9】





【図10】



フロントページの続き

(71)出願人 597067574

201 BROADWAY, CAMBRIDGE,  
MASSACHUSETTS  
02139, U. S. A.

(72)発明者 ルウシェン・ペン

アメリカ合衆国、カリフォルニア州、サン・ホセ、サン・トーマス・アキノ・ロード 129、アパートメント・ナンバー 211

(54)【発明の名称】 複数のサーバでオブジェクトを同期させる汎用システム、2つのサーバでオブジェクトを同期させるシステム、2つのサーバでオブジェクトを同期させる方法、2つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期される汎用同期システム、2つのサーバでのオブジェクトの同期における競合を検出して解消する方法